

## Syllabus

<b>F.Y.B.Sc. (IT) Semester - II</b>		
<b>Subject : Web designing and Programming</b>		
<b>Unit - I</b>	<p><b>Internet and WWW</b> : What is Internet?, Introduction to internet and its applications, E-mail, telnet, FTP, e-commerce, video conferencing, e-business. Internet service providers, domain name server, internet address World Wide Web (WWW) : World Wide Web and its evolution, uniform resource locator (URL), browsers - internet explorer, netscape navigator, opera, firefox, chrome, mozilla. Search engine, web saver - apache, IIS, proxy server, HTTP protocol.</p>	<b>8 lectures</b>
<b>Unit - II</b>	<p><b>HTML and Graphics</b> : HTML Tag Reference, Global Attributes, Event Handlers, Document Structure Tags, Formatting Tags, Text Level formatting, Block Level formatting, List Tags, Hyperlink tags, Image and Image maps, Table tags, Form Tags, Frame Tags, Executable content tags.</p> <p><b>Imagemaps</b> : What are Imagemaps? Client-side Imagemaps, Server-side Imagemaps, Using Server-side and Client-side Imagemaps together, alternative text for Imagemaps,</p> <p><b>Tables</b> : Introduction to HTML tables and their structure, The table tags, Alignment, Aligning Entire Table, Alignment within a row, Alignment within a cell, Attributes, Content Summary, Background color, Adding a Caption, Setting the width, Adding a border, Spacing within a cell, Spacing between the cells, spanning multiple rows or columns, Elements that can be placed in a table, Table Sections and column properties, Tables as a design tool</p> <p style="text-align: center;">F.Y.B.Sc. (IT) Semester - II</p> <p><b>Frames</b> : Introduction to Frames, Applications, Frames document, The &lt;FRAMESET&gt; tag, Nesting &lt;FRAMESET&gt; tag, Placing content in frames with the &lt;FRAME&gt; tag, Targeting named frames, Creating floating frames, Using Hidden frames,</p>	<b>8 lectures</b>

	<p><b>Forms</b> : Creating Forms, The &lt;FORM&gt; tag, Named Input fields, The &lt;INPUT&gt; tag, Multiple lines text windows, Drop down and list boxes, Hidden, Text, Text Area, Password, File Upload, Button, Submit, Reset, Radio, Checkbox, Select, Option, Forms and Scripting, Action Buttons, Labelling input files, Grouping related fields, Disabled and read-only fields, Form field event handlers, Passing form data</p> <p><b>Style Sheets</b> : What are style sheets?, Why are style sheets valuable? Different approaches to style sheets, Using Multiple approaches, Linking to style information in s separate file, Setting up style information, Using the &lt;LINK&gt; tag, embedded style information, Using &lt;STYLE&gt; tag, Inline style information</p>	
<p><b>Unit - III</b></p>	<p><b>Java Script</b> : Introduction, Client-Side JavaScript, Server-Side JavaScript, JavaScript Objects, JavaScript Security,</p> <p><b>Operators</b> : Assignment Operators, Comparison Operators, Arithmetic Operators, % (Modulus), ++ (Increment), -- (Decrement), -(Unary Negation), Logical Operators, Short-Circuit Evaluation, String Operators, Special Operators, ? (Conditional operator), ,(Comma operator), delete, new, this, void</p> <p><b>Statements</b> : Break, comment, continue, delete, do ... while, export, for, for...in, function, if...else, import, labelled, return, switch, var, while, with,</p> <p><b>Core JavaScript (Properties and Methods of Each)</b> : Array, Boolean, Date, Function, Math, Number, Object, String, regExp</p> <p><b>Document and its associated objects</b> : document, Link, Area, Anchor, Image, Applet, Layer</p> <p><b>Events and Event Handlers</b> : General Information about Events, Defining Event Handlers, event, onAbort, onBlur, onChange, onClick, onDbIclick, onDragDrop, onError, onFocus, onKeyDown, onKeyPress, onKeyUp, onLoad, onMouseDown, onMouseMove, onMouseOut, onMouseOver, onMouseUp, onMove, onReset, onResize, onSelect, onsubmit, onUnload</p>	<p><b>8 lectures</b></p>

<b>Unit- IV</b>	<b>XML</b> : Introduction to XML, Anatomy of an XML, document, Creating XML Documents, Creating XML DTDs, XML Schemas, XSL	<b>8 lectures</b>
<b>Unit-V</b>	<b>PHP</b> : Why PHP and MySQL?, Server-side web scripting, Installing PHP, Adding PHP to HTML, Syntax and Variables, Passing information between pages, Strings, Arrays and Array Functions, Numbers, Basic PHP errors / problems.	<b>8 lectures</b>
<b>Unit-VI</b>	<b>Advanced PHP and MySQL</b> : PHP/MySQL Functions, Displaying queries in tables, Building Forms from queries, String and Regular Expressions, Sessions, Cookies and HTTP, Type and Type Conversions, E-Mail	<b>8 lectures</b>

**Reference :**

1. Web Design The complete Reference, Thomas Powell, Tata McGrawHill
2. HTML and XHTML The complete Reference, Thomas Powell, Tata McGrawHill
3. JavaScript 2.0 : The Complete Reference, Second Edition by Thomas Powell and Fritz Schneider
4. PHP : The Complete Reference By Steven Holzner, Tata McGrawHill

**Term Work for USIT201**

- i) Assignments : Should contain at least 2 assignments covering the Syllabus.
- ii) Class Tests : One. Also Known as Unit Test or In-Semester Examinations
- iii) Tutorial : Minimum Three tutorials covering the syllabus

**Practicals (USIT2P1) :**

<b>Journal Practical</b>	<b>3 lectures per Week (1 Credit)</b>
<p><b>List of Practical</b></p> <ol style="list-style-type: none"><li>1. Design a web page using different text formatting tags.</li><li>2. Design a web page with links to different pages and allow navigation between pages.</li><li>3. Design a web page with Imagemaps.</li><li>4. design a web page with different tables. Design a webpage suing table so that the content appears well placed.</li><li>5. Design a webpage using frames.</li><li>6. Design a web page with a form that uses all types of controls.</li><li>7. Design a website using style sheets so that the pages have uniform style.</li><li>8. Using Java Script design a web page that prints factorial / Fibonacci series / any given series.</li><li>9. Design a form with a test box and a command button. Using Java Script write a program whether the number entered in the text box is a prime number or not.</li><li>10. Design a form and validate all the controls placed on the form using Java Script.</li><li>11. Design a DTD, corresponding XML document and display it in browser using CSS.</li><li>12. Design an XML document and display it in browser using XSL.</li><li>13. Design XML Schema and corresponding XML document.</li></ol>	



## INTERNET AND WWW

### Unit Structure

- 1.0 Objective
- 1.1 Introduction to internet and its applications.
  - 1.1.1 E mail
  - 1.1.2 Telnet
  - 1.1.3 FTP
  - 1.1.4 E-commerce
  - 1.1.5 Video conferencing
  - 1.1.6 E business.
- 1.2 Internet service providers
- 1.3 Domain name servers
- 1.4 Internet address
- 1.5 World wide web and its evolution
- 1.6 URL
- 1.7 Browsers
  - 1.7.1 Internet explorer,
  - 1.7.2 netscape navigator,
  - 1.7.3 opera,
  - 1.7.4 firefox,
  - 1.7.5 chrome,
  - 1.7.6 Mozilla,
- 1.8 Search Engine
- 1.9 Web server
  - 1.9.1 Apache
  - 1.9.2 IIS
  - 1.9.3 Proxy Server
- 1.10 HTTP protocol.
- 1.11 Summery
- 1.12 Unit End exercise

---

## **1.0 OBJECTIVE:**

---

After reading through this chapter, you will be able to –

- Understand concept of Internet and world wide web, their applications.
- List the services provided by Internet Service providers with examples.
- Define domain name server and list various domains.
- Understand the concept of Internet address.
- Understand the function of a URL and web browsers.
- Use different web browsers.
- Use the search engines to search for required information over the internet.
- Understand the need and use of a web server and proxy server.

---

## **1.1 INTRODUCTION TO INTERNET AND ITS APPLICATIONS**

---

*Internet is ---*

- A computer network consisting of a worldwide network of computers that use the TCP/IP network protocols to facilitate data transmission and exchange.
- The Internet is a global system of interconnected computer networks that use the standard Internet Protocol Suite (TCP/IP) to serve billions of users worldwide.
- The term Internet actually refers to the combined collection of academic, commercial, and government networks connected over international telecommunication backbones and routed using IP addressing.

The internet has gained popularity rapidly as it is used for various purposes. Few of the main applications of internet are listed below –

*Applications of Internet ---*

### **1.1.1 E mail (Electronic mail)**

- Electronic mail (also known as email or e-mail) is one of the most commonly used services on the Internet, allowing people to send messages to one or more recipients.

- Email was invented by Ray Tomlinson in 1972.
- Electronic mail is a method of exchanging digital messages from an author to one or more recipients.
- Modern email operates across the Internet or other computer networks. Today's email systems are based on a store-and-forward model.
- Email servers accept, forward, deliver and store messages. Neither the users nor their computers are required to be online simultaneously; they need connect only briefly, typically to an email server, for as long as it takes to send or receive messages.
- An email message consists of three components, the message *envelope*, the message *header*, and the message *body*.
- Header contains information about who sent the message, the recipient(s) and the route.
- Header also usually contains descriptive information, such as a subject header field and a message submission date/time stamp.
- Email message body contains text (7bit ASCII) as well as multimedia messages. These processes are declared in Multipurpose Internet Mail Extensions (MIME). MIME is set of RFCs (Request for Comment)
- Network based emails are exchanged over the internet using the SMTP (Simple Mail Transfer protocol).
- In the process of transporting email messages between systems, SMTP communicates delivery parameters using a message *envelope* separate from the message (header and body) itself.
- Email addresses (both for senders and recipients) are two strings separated by the character "@" (the "at sign"): such as `user@domain`
- The right-hand part describes the domain name involved, and the left-hand part refers to the user who belongs to that domain.
- An email address can be up to 255 characters long and can include the following characters:
  - Lowercase letters from a to z;
  - Digits
  - The characters ".", "\_" and "-" (full stop, underscore, and hyphen)
 In practice, an email address often looks something like this:  
[fname.lname@provider.domain](mailto:fname.lname@provider.domain)

### 1.1.2 telnet

- Telnet is a network protocol used in any network (internet or LAN) for bidirectional text oriented communication.
- telnet standard was defined in 1973, before which it was considered as adhoc protocol.
- Original purpose of the telnet protocol was to login to the remote computers on the network.
- telnet protocol uses 'virtual terminal' to connect to the remote hosts.
- Virtual terminal is a application service that allows host in a multi terminal network to communicate with other hosts irrespective of terminal type or characteristics.
- Telnet uses the TCP protocol for transmission of 8 byte data.
- Most network equipment and operating systems with a TCP/IP stack support a Telnet service for remote configuration (including systems based on Windows NT)
- The term telnet may also refer to the software that implements the client part of the protocol.
- telnet is a client server protocol, which is based upon reliable connection oriented communication transport and basic use of telnet is to make a connection to the TCP protocol.
- Data transferred over telnet is vulnerable as telnet does not use any encryption technique to mask or protect the data.
- Most implementations of Telnet have no authentication that would ensure communication is carried out between the two desired hosts and not intercepted in the middle.
- Commonly used Telnet daemons have several vulnerabilities discovered over the years.
- Extensions to the Telnet protocol provide Transport Layer Security (TLS) security and Simple Authentication and Security Layer (SASL) authentication that address the above issues.
- Few applications of telnet include the 'putty' TCP client which can access a linux server using windows operating system, Absolute telnet (windows client) and RUMBA (terminal emulator).

### 1.1.3 FTP

- File transfer protocol is a simple and standard network protocols that transfers a file from one host to the other over a TCP network.
- Based on client server architecture.



- Utilizes separate control and data connection for client and server to transmit and receive file(s) over the network.
- It is an application protocol that uses the internet's TCP/IP suite.
- Mainly used to transfer the web pages or related data from the source or creator to a host that acts as a server to make the page or file available to other hosts (uploading) or downloading programs and other files from server to a host.
- FTP protocol can perform over a active or passive connection.
- When a connection is made from the client to server, it is called as control connection and it remains open for duration of session. This connection is responsible for establishing connectivity between client and server.
- Other connection opened by client (passive) or server (active) is called data connection and is used to transfer the data.
- As separate ports are used by client and server for these connections, FTP becomes an put of band protocol.
- Data transfer can take place in following three modes
  - Stream mode : data is sent in a continuous stream where FTP does not do any formatting.
  - Block mode: FTP breaks the data into several blocks (block header, byte count, and data field) and then passes it on to TCP.
  - Compressed mode: Data is compressed using a single algorithm.
- FTP is a old protocol and is basically low in security aspect. Data transferred over FTP is not encrypted and is in clear text format. Hence the data like usernames, passwords can be read by anyone who can capture the FTPed package. Newer versions of the protocol, however, apply secure shell protocol (SSH) and avoid all the problems faced by FTP.
- Following are few types of FTP protocol with additional features
  - Anonymous FTP : Users login using an 'anonymous' account to protect their confidential data.
  - Remote FTP: FTP commands executed on a remote FTP server
  - FTP with web browser and firewall support.
  - Secure FTP (SFTP, FTPS)

#### **1.1.4 E Commerce**

- Electronic commerce can be defined as use of electronic communications, particularly via the internet, to facilitate the purchase/sale of goods and services. E-commerce includes all

forms of electronic trading including electronic data interchange (EDI), electronic banking, electronic mail and other online services.

- E transactions are of two categories. – virtual products like policies and actual retail products.
- Most of e transactions of actual products involve physical transportation of goods which are purchased over the electronic media.
- Online retailing has gained a name of E tailing.
- Electronic commerce is generally considered to be the sales aspect of e-business. It also consists of the exchange of data to facilitate the financing and payment aspects of the business transactions.
- Originally, electronic commerce was identified as the facilitation of commercial transactions electronically, using technology such as Electronic Data Interchange (EDI) and Electronic Funds Transfer (EFT).
- Other forms of e commerce were established with the growth and use of credit cards, and air line reservation system going online.
- Electronic commerce of the modern era (post 1990) includes technologies like enterprise resource planning (ERP), data warehousing and data mining.
- The electronic transactions between two businesses like dealer and wholesaler or wholesaler and retailer come in the B2B (business to business) E commerce category.
- Other popular E commerce categories would be business to consumer (B2C) and business to government (B2G)
- Volume of B2B transaction is much higher as compared to the volume of B2C transactions. Reason for this is, many transactions at B2B level lead to finished good and this leads to just one B2C transaction.
- In an example, if a customer buys a product, say a pen, that would be a B2C transaction. But the transaction leading this one, including purchase of plastic, ink, refill, moulds etc would be B2B transaction. Also the sale of the pen to the retailer by the manufacturing company, like cello, is B2B transaction.
- Other form of B2C transactions are business to individual, where the record of an individual's transaction is maintained.
- C2C is consumer to consumer, or citizen to citizen E commerce. Here customers can perform transaction via a third party. Like a product can be posted on amazon.com and will be sold to another consumer through amazon.

- C2B E commerce model is reverse of traditional business to consumer approach. This can be explained by a internet blog or a social networking site where author can have a link in his blog article to online sale of a product (promoting the business). This has become possible due to advancements in technology and reduced costs of technology.
- Unique attribute of e commerce is negotiation facility and its immediate results. Also, in E commerce transactions, integration of transactions is automated.

### **1.1.5 Video Conferencing**

- Video conferencing or video teleconference is a set of telecommunication technologies which allow one or more locations to transmit and receive video and audio signals simultaneously.
- This is known as visual collaboration.
- Simple analog video conferencing is achieved by two closed circuit television systems connected with coaxial cables or radio waves.
- This type of communication was established from 1968.
- Modern video conferencing is IP based and through more efficient video compression technologies, allowing desktop or PC based video conferencing.
- Videotelephony is now popular due to free internet services.
- Core technology used for this is compression of audio and video signals. Hardware and software used for this task is called as codec (coder/ decoder). Compression rate achieved is almost 1:500. The resultant stream of binary data is sent in packet form through digital networks.
- The components required for a videoconferencing system include:
  - Video input : video camera or webcam
  - Video output: computer monitor , television or projector
  - Audio input: microphones, CD/DVD player, cassette player, or any other source of PreAmp audio outlet.
  - Audio output: usually loudspeakers associated with the display device or telephone
  - Data transfer: analog or digital telephone network, LAN or Internet
- There are basically two types of videoconferencing systems.

- Dedicated systems: all required components (i.e. software and hardware based codec, control computer and video camera, electrical interfaces) packed in a single console application. They include large group, small group, portable and non portable video conferencing systems.
- Desktop Systems: add ons to normal computing systems transforming these systems to videoconferencing devices.
- There are following layers in the videoconferencing technology –
  - User interface
  - Conference control
  - Control or signal plane
  - Media plane.
- Videoconferencing has following modes
  - Voice activated switch
  - Continuous presense
- Problems faced by videoconferencing
  - Echo: echo is defined as reflected source wave interference with new wave created by the source. i.e. signal coming out from the source interferes with newly coming source and generating unwanted input signal. This may result into remote party receiving their own sounds again. This can be avoided by using an algorithm called as AEC (Acoustic Echo cancellation).
  - Eye contact : in videoconferencing, due to time delays and parallax, communicators have a feel of the other party avoiding eye contact and can result into issues in professional communication. This can be avoided by using special stereo reconstruction technique.
  - Signal latency: The information transport of digital signals in many steps needs time. In a telecommunicated conversation, an increased latency larger than about 150-300ms becomes noticeable and is soon observed as unnatural and distracting. Therefore, next to a stable large bandwidth, a small total round-trip time is another major technical requirement for the communication channel for interactive videoconferencing.

#### **1.1.6 E business**

- E business is conduct of business over the internet, which includes buying and selling of goods and even services.
- In other words it is application of information and communication technologies in support of all activities in business.

- Applications of E business are divided into following categories –
  - Internal business systems --
    - Customer Relationship Management (CRM)
    - Enterprise Resource Planning (ERP)
    - Human resource management system(HRMS)
  - Enterprise Communication and collaboration
    - Content management
    - E- mails
    - Voice mails
    - Web conferencing
  - Electronic commerce
    - B2B (business to business)
    - B2C (business to customer)
    - B2E business-to-employee
    - B2G business-to-government
    - G2B government-to-business
    - G2G (government-to-government)
    - G2C (government-to-citizen )
    - C2C (consumer-to-consumer )
    - C2B (consumer-to-business)
- A business model is defined as the organization of product, service and information flows, and the source of revenues and benefits for suppliers and customers. The concept of e-business model is the same but used in the online presence.
- Few e business models are –
  - E-shops
  - E-commerce
  - E-procurement
  - E-malls
  - E-auctions
  - Virtual Communities
- E business has more security risks as compared to a regular business, as E business has many more users at a time. Keeping the large information confidential is a difficult task. Also, data integrity, authenticity and storage of data are some challenges faced by E business.
- Some methods to provide security are physical security as well as encryption in data storage, transmission, antivirus software and firewalls. Digital signature is another way to confirm the ownership of a document.

---

## **1.2 INTERNET SERVICE PROVIDER**

---

- An Internet service provider (ISP) is a company that provides access to the Internet, hosts data, or does both. ISP is also known as IAP

(internet access provider) Access ISPs connect customers to the Internet using copper, wireless or fibre connections. Hosting ISPs lease server space for smaller businesses and host other people servers (colocation). Transit ISPs provide large tubes for connecting hosting ISPs to access ISPs.

- As internet gained popularity, it was essential to provide internet access to many people or many hosts. Due to the increasing demand to access internet, commercial ISP came into existence in 1990.

***Technologies used –***

For users and small business applications -

- Dial up connection
- DSL (digital subscriber line)
- Broadband wireless connection
- Cable modem
- Fibre optical connection

For medium to large businesses or for other ISPs,

- DSL
- Ethernet
- Metro Ethernet
- Gigabyte Ethernet
- Frame relay
- Satellite Ethernet

- ***ISP connections –***

ISPs which provide connections through phone lines like dial ups, do not seek any information about the caller's (user of the internet) physical location or address. So, caller from any location which is in reach of the ISP, can use the services provided.

Other way of getting connected through ISP is using cable or any other lines. Here, fixed registration of the user at the ISP side is essential.

- ***Services provided –***

ISP host usually provide e mail, FTP and web hosting services. Other services can be like virtual machines, clouds or entire physical servers where clients can run their own softwares.

ISPs often take services from their upstream ISPs. i.e. they work in hierarchy. The ISPs are divided into three categories –

- **Peering** : ISPs taking services from upstream and getting connected to each other to exchange data and to control network traffic through peering points, or internet exchange points. These

points help save one more upstream ISP and cut down on the cost. The ISPs which do not need upstream ISP are called Tier 1 ISP.

- Virtual ISP (VISP) : this is an ISP which purchases services from other ISP and gives them to the end user, without any set up of its own.
- Free ISP: these are ISPs which provide services free of cost to the users and display advertisements till the users are connected. These are called as freenets. These are normally run on no profit basis.

---

### 1.3 DOMAIN NAME SERVERS

---

- *Domain Name System is –*

DNS is part of a domain name system. It is hierarchical naming system built on a distributed database for resource connected to the internet or a private network. The main purpose of this system is to translate domain names meaningful to humans into names or rather numeric streams which help the corresponding network devices to identify the resource or domain. Domain name system makes it possible to give or allot names to domains or group of networks irrespective of their physical locations.

- *Domain Name Server is –*
- Domain name system assigns domain name servers for allotting names and mapping these names to IP addresses. In other words domain name servers are nodes of the domain name system which acts like a client server system. Each domain has at least one authoritative DNS server that publishes information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root nameservers, the servers to query when looking up (resolving) a TLD (Top level domain). Authoritative DNS can either be a master or a slave. Master DNS keeps record of all zone records. Slave DNS uses a automatic update mechanism to maintain copies of records of its master. Every top level domain requires a primary DNS and at least one secondary DNS. Every DNS query must start with recursive queries at the root zone, for authoritative DNS.
- To improve efficiency, reduce DNS traffic across the Internet, and increase performance in end-user applications, the Domain Name System supports DNS cache servers which store DNS query results for a period of time determined in the configuration (time-to-live) of the domain name record in question.
- The client-side of the DNS is called a DNS resolver. It is responsible for initiating and sequencing the queries that ultimately lead to a full resolution (translation) of the resource sought, e.g., translation of a domain name into an IP address.

- A DNS query may be either a non-recursive query or a recursive query
- The resolver, or another DNS server acting recursively on behalf of the resolver, negotiates use of recursive service using bits in the query headers.
- Resolving usually entails iterating through several name servers to find the needed information. However, some resolvers function simplistically and can communicate only with a single name server. These simple resolvers (called "stub resolvers") rely on a recursive name server to perform the work of finding information for them.
- **Operation of DNS –**
  - Domain name resolvers determine the appropriate domain name servers responsible for the domain name in question by a sequence of queries starting with the right-most (top-level) domain label.
  - DNS recorsor consults three name servers to resolve one address. The process is as follows –
    - A network host is configured with an initial cache (so called *hints*) of the known addresses of the root nameservers. Such a *hint file* is updated periodically by an administrator from a reliable source.
    - A query to one of the root servers to find the server authoritative for the top-level domain.
    - A query to the obtained TLD server for the address of a DNS server authoritative for the second-level domain.
    - Repetition of the previous step to process each domain name label in sequence, until the final step which returns the IP address of the host sought.

---

## 1.4 INTERNET ADDRESS

---

- Internet address follows the TCP/IP suite hence, it is also known as the IP address.
- Internet address has a job of identifying a node on the network. In oher words, it is a numeric lable attached to every system (computer or any other device). The basic function of IP address are two –
- Identification of computer or node or device and location addressing.
- The designers of the Internet Protocol defined an IP address as a 32-bit number[1] and this system, known as Internet Protocol Version 4 (IPv4), is still in use today. However, due to the enormous growth of the Internet and the predicted depletion of available addresses, a new addressing system (IPv6), using 128 bits for the address, was developed in 1995,[3] standardized as RFC 2460 in 1998,[4] and is being deployed worldwide since the mid-2000s.



- IP addresses are binary numbers, but they are usually stored in text files and displayed in human-readable notations, such as 172.16.254.1 (for IPv4)
- IPv4 address is a 32 bit number, which uses the decimal dotted notation consisting of 4 decimal numbers each ranging from 0 to 255 separated by dots. Network administration divides the IP address into two parts. – the most significant 8 bits are called network address portion the remaining bits are known as rest bits or host bits or identifiers and they are used for host numbering in a network.
- Although IPv4 provides 4.3 billion addresses, they are exhausted due to high demand and as a result, insufficient addresses available with IANA (Internet assigned numbers authority). The primary address pool of IANA is expected to get exhausted by mid 2011. To permanently address the problem, new version of IP i.e. IPv6 was brought forward, this version moved the size of IP address from 32 bit to 128 bits.
- Both IPv4 as well as IPv6 have reserved addresses for private or internal networks. This is termed as private addressing.
- Both IPv4 and IPv6 have subnetting effect. That mean, IP networks can be divided into smaller groups or subnets. IP addresses two constituents that is network address and host identifier or interface identifier are used for this purpose.
- Internet Protocol addresses are assigned to a host either anew at the time of booting, or permanently by fixed configuration of its hardware or software. Persistent configuration is also known as using a *static IP address*. In contrast, in situations when the computer's IP address is assigned newly each time, this is known as using a *dynamic IP address*

---

## 1.5 WORLD WIDE WEB AND ITS EVOLUTION

---

- The World Wide Web, abbreviated as WWW or W3 and commonly known as the Web, is a system of interlinked hypertext documents accessed via the Internet.
- With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks.
- The World-Wide Web was developed to be a pool of human knowledge, and human culture, which would allow collaborators in remote sites to share their ideas and all aspects of a common project.

### *Evolution of WWW*

- In March 1989, Tim Berners-Lee wrote a proposal that referenced ENQUIRE, a database and software project he had built in 1980, and described a more elaborate information management system.
- on November 12, 1990, with help from Robert Cailliau, Tim Berners-Lee published a more formal proposal to build a "Hypertext project" called "WorldWideWeb" (one word, also "W3") as a "web" of "hypertext documents" to be viewed by "browsers" using a client-server architecture.
- This proposal estimated that a read-only web would be developed within three months and that it would take six months to achieve "the creation of new links and new material by readers, to achieve universal authorship!" as well as "the automatic notification of a reader when new material of interest to him/her has become available."
- A NeXT Computer was used by Berners-Lee as the world's first web server and also to write the first web browser, WorldWideWeb, in 1990.
- Tools needed were a working Web the first web browser (which was a web editor as well); the first web server; and the first web pages, which described the project itself.
- On August 6, 1991, Tim Berners-Lee posted a short summary of the World Wide Web project on the alt.hypertext newsgroup.
- This date also marked the debut of the Web as a publicly available service on the Internet. The first photo on the web was uploaded by Berners-Lee in 1992, an image of the CERN house band Les Horribles Cernettes.
- The first server outside Europe was set up at SLAC to host the SPIRES-HEP database in 91 – 92.
- The concept of hypertext originated with older projects from the 1960s, such as the Hypertext Editing System (HES) at Brown University by Ted Nelson and Douglas Engelbart.
- Tim Berners Lee introduced the concept of the Universal Document Identifier (UDI), later known as Uniform Resource Locator (URL) and Uniform Resource Identifier (URI); the publishing language HyperText Markup Language (HTML); and the Hypertext Transfer Protocol (HTTP).

- In 1993, a graphical browser was developed by a team at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign (NCSA-UIUC), led by Marc Andreessen. This was the first web browser ever.

---

## 1.6 URL

---

- Uniform Resource Locator (URL) is a Uniform Resource Identifier (URI) that specifies where an identified resource is available and the mechanism for retrieving it.
- An example of the use of URLs is the addresses of web pages on the World Wide Web, such as *http://www.example.com/*.
- The format is based on Unix file path syntax, where forward slashes are used to separate directory or folder and file or resource names.
- Conventions already existed where server names could be prepended to complete file paths, preceded by a double-slash
- Every URL consists of some of the following:
  - The scheme name (commonly called protocol), followed by a colon. The scheme name defines the namespace, purpose, and the syntax of the remaining part of the URL.
  - Domain Name depending upon scheme(alternatively, IP address). The domain name or IP address gives the destination location for the URL.
  - An optional port number; if omitted, the default for the scheme is used
  - Path of the resource to be fetched or the program to be run. The path is used to specify and perhaps find the resource requested. It may be case-sensitive for non window based servers. Eg: *http://www.mudlle.ac.in/news.html*
  - A query string for scripts The query string contains data to be passed to software running on the server. It may contain name/value pairs separated by ampersands, for example *?first\_name=John&last\_name=Doe*.
  - Optional fragment identifier that specifies a part or a position within the overall resource or document. When used with HTTP, it usually specifies a section or location within the page, and the browser may scroll to display that part of the page.

- When resources contain references to other resources, they can use relative links to define the location of the second resource.
- relative URLs are dependent on the original URL containing a hierarchical structure against which the relative link is based.
- the ftp, http, and file URL schemes are examples of some that can be considered hierarchical, with the components of the hierarchy being separated by "/"
- A URL is a URI that, "in addition to identifying a resource, provides a means of locating the resource by describing its primary access mechanism.

---

## 1.7 BROWSERS

---

- A web browser or Internet browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.
- Web browsers can also be used to access information provided by Web servers in private networks or files in file systems. Some browsers can also be used to save information resources to file systems.
- Primary function of a browser is to identify the URI and brings the information resource to user.
- This process begins when user inputs the URI in the browser. Prefix of the URI describes how to interpret the URI. Most URIs have resource retrieved over Hyper text Transfer Protocol. Some web browsers also support prefixes like FTP.
- Once this is done, the HTML script is passed to the browser's layout engine. To make the script interactive java script support is needed. With this, browser can interpret text, images, video and interactive scripts.
- All major browsers allow users to access multiple information resources at the same time in different windows or in tabs. Major browsers include pop up blockers to prevent windows to open without users consent.
- Most major web browsers have these user interface elements in common:
  - Back and forward buttons to go back to the previous resource and forward again.

- A history list, showing resources previously visited in a list (typically, the list is not visible all the time and has to be summoned)
- A refresh or reload button to reload the current resource.
- A stop button to cancel loading the resource. In some browsers, the stop button is merged with the reload button.
- A home button to return to the user's home page
- An address bar to input the Uniform Resource Identifier (URI) of the desired resource and display it.
- A search bar to input terms into a search engine
- A status bar to display progress in loading the resource and also the URI of links when the cursor hovers over them, and page zooming capability.
  - The usage share of web browsers is as shown below. (Source: Median values )
- Internet Explorer (43.55%)
- Mozilla Firefox (29.0%; Usage by version number)
- Google Chrome (13.89%)
- Safari (6.18%)
- Opera (2.74%)
- Mobile browsers (4.45%)
  - Some special web browsers are listed below –

### **1.7.1 INTERNET EXPLORER**

- Windows Internet Explorer (formerly Microsoft Internet Explorer, commonly abbreviated IE or MSIE) is a series of graphical web browsers developed by Microsoft and included as part of the Microsoft Windows line of operating systems starting in 1995.
- It was first released as part of the add-on package Plus! for Windows 95 that year. Later versions were available as free downloads, or in service packs. It was part of later versions of windows operating systems.
- The latest stable release is Internet Explorer 9, which is available as a free update for Windows 7, Windows Vista, Windows Server 2008 and Windows Server 2008 R2.
- Internet Explorer uses a componentized architecture built on the Component Object Model (COM) technology. It consists of several

major components, each of which is contained in a separate Dynamic-link library (DLL) and exposes a set of COM programming interfaces hosted by the Internet Explorer main executable, 'iexplore.exe'

- Internet Explorer uses a zone-based security framework that groups sites based on certain conditions, including whether it is an Internet- or intranet-based site as well as a user-editable whitelist. Security restrictions are applied per zone; all the sites in a zone are subject to the restrictions.

### 1.7.2 NETSCAPE NAVIGATOR

- Netscape Navigator is a proprietary web browser that was popular in the 1990s. It was the most popular web browser till 2002, after which competitor browsers have taken over the market of netscape.
- Netscape Navigator was based on the Mosaic web browser.
- Netscape announced in its first press release (October 13, 1994) that it would make Navigator available without charge to all non-commercial users, and Beta versions of version 1.0 and 1.1 were indeed freely downloadable in November 1994 and March 1995, with the full version 1.0 available in December 1994.
- The first few releases of the product were made available in "commercial" and "evaluation" versions.
- During development, the Netscape browser was known by the code name Mozilla. Mozilla is now a generic name for matters related to the open source successor to Netscape Communicator.

### 1.7.3 OPERA

- Opera is a web browser and Internet suite developed by Opera Software. The browser handles common Internet-related tasks such as displaying web sites, sending and receiving e-mail messages, managing contacts, chatting on IRC, downloading files via BitTorrent, and reading web feeds.
- Opera is offered free of charge for personal computers and mobile phones. This is the most popular mobile phone browser and is not packages in desktop operating system.
- Features include tabbed browsing, page zooming, mouse gestures, and an integrated download manager. Its security features include built-in phishing and malware protection, strong encryption when browsing secure websites, and the ability to easily delete private data such as HTTP cookies.

- Opera runs on a variety of personal computer operating systems, including Microsoft Windows, Mac OS X, Linux, and FreeBSD
- Opera includes built-in tabbed browsing, ad blocking, fraud protection, a download manager and BitTorrent client, a search bar, and a web feed aggregator. Opera also comes with an e-mail client called Opera Mail and an IRC chat client built in.
- Opera has several security features visible to the end user. One is the option to delete private data, such as HTTP cookies, the browsing history, and the cache, with the click of a button. This lets users erase personal data after browsing from a shared computer.
- Opera Mobile is an edition of Opera designed for smartphones and personal digital assistants (PDAs)

#### **1.7.4 MOZILLA FIREFOX**

- Mozilla Firefox is a free and open source web browser descended from the Mozilla Application Suite and managed by Mozilla Corporation. As of February 2011[update], Firefox is the second most widely used browser with approximately 30% of worldwide usage share of web browsers.
- To display web pages, Firefox uses the Gecko layout engine, which implements most current web standards.
- The latest Firefox features[15] include tabbed browsing, spell checking, incremental find, live bookmarking, a download manager, private browsing, location-aware browsing (also known as "geolocation") based exclusively on a Google service.
- Firefox runs on various operating systems including Microsoft Windows, Linux, Mac OS X, FreeBSD, and many other platforms.

#### **1.7.5 CHROME**

- Chrome, the web browser by Google, is rapidly becoming popular due to following features-
  - SPEED: Chrome is designed to be fast in every possible way: It's quick in starting up from the desktop, loading web pages and running complex web applications.
  - SIMPLICITY: Chrome's browser window is streamlined, clean and simple. Chrome also includes features that are designed for efficiency and ease of use. For example, you can search and navigate from the same box, and arrange tabs however you wish.
  - SECURITY: Chrome is designed to keep you safer and more secure on the web with built-in malware and phishing protection, autoupdates to make sure the browser is up-to-date with the latest security updates, and more. Learn more about Chrome's security features.

- Chrome is the first browser to incorporate machine translation in the browser itself, without requiring additional plugins or extensions.

### 1.7.6 MOZILLA

---

## 1.8 SEARCH ENGINE

---

- A web search engine is designed to search for information on the World Wide Web and FTP servers. The search results are generally presented in a list of results and are often called hits. The information may consist of web pages, images, information and other types of files. Some search engines also mine data available in databases or open directories.
- The very first tool used for searching on the Internet was [Archie](#).
- The first web robot, the Perl-based World Wide Web Wanderer was built and used by it to generate an index called 'Wandex'. The purpose of the Wanderer was to measure the size of the World Wide Web.
- Around 2000, Google's search engine rose to prominence. The company achieved better results for many searches with an innovation called PageRank. This iterative algorithm ranks web pages based on the number and PageRank of other web sites and pages that link there, on the premise that good or desirable pages are linked to more than others.
- Web search engines work by storing information about many web pages, which they retrieve from the html itself. These pages are retrieved by a Web crawler (sometimes also known as a spider) — an automated Web browser which follows every link on the site.
- This information is then analyzed and indexed. The contents of each page are then analyzed to determine how it should be indexed. The purpose of an index is to allow information to be found as quickly as possible.



---

## **1.9 WEB SERVER**

---

1.9.1 APACHE

1.9.2 IIS

1.9.3 PROXY SERVER

---

## **1.10 HTTP PROTOCOL.**

---

---

## 1.11 SUMMERY

---

---

## 1.12 EXERCISE

---

1. Define the internet. What protocol suit does it follow?
2. What is email? How is it sent and received?
3. Describe the three components of email.
4. What is meant by email address? What are the required parts of email address?
5. Are following email addresses valid?
  - a. 124sir@idol.com
  - b. 11 myname@yahoo.org
  - c. Seema\_Sathye@ server.co.in
  - d. Piyush\_mishra@myservices.net.in
6. What is telnet used for?
7. What is virtual terminal? What is it used for?
8. Name applications of telnet.
9. Why is FTP protocol used?
10. Explain different connections that can be used by FTP.
11. What are drawbacks of telnet and FTP?
12. Define E commerce. What are the advantages of e commerce?
13. Give one example of B2B, B2C and C2C e commerce.
14. Identify following E commerce category –
  - a. Sale of online admission form for a college.
  - b. Submission of the above form.
  - c. Online resale of a second hand car.
  - d. Purchase of raw material by an automobile company.

15. What is visual collaboration?
16. What is codec how does it function?
17. What are different types of video conferencing system?
18. List the components of video conferencing system.
19. Discuss the problems faced by video conferencing system.
20. Define E business.
21. List a few applications of e business.
22. What is a E business model? Give three examples of E business model.
23. What are risks for E business? What are the solutions available for these risks?
24. What is ISP? Explain the role of ISP in an internet connection.
25. Classify following technologies of ISP in business or home connections
  - a. A dial up connection with speed 1Mbps
  - b. A connection to a LAN using leased cable lines
  - c. Hosting of personal web page
  - d. Use of wi-fi for a laptop.
26. What are different services provided by ISP?
27. Explain what is peering? What is its advantage?
28. What is VISIP? Give one example to explain its use.
29. Discuss the concept of freenet and its importance.
30. What are domain name servers? What is their function?
31. What is internet address? How is it assigned?
32. Write a note on evolution of www
33. What is a web browser? List and compare different available web browsers



## HTML AND GRAPHICS

### Unit Structure

- 2.0. Objective
- 2.1. What is HTML?
- 2.2. Global Attributes
- 2.3. Event Handlers
- 2.4. Document Structure Tags
- 2.5. Formatting Tags
- 2.6. List Tags
- 2.7. Hyperlink Tags
- 2.8. Image and Imagemaps
- 2.9. Table Tags
- 2.10. Form Tags
- 2.11. Frame Tags
- 2.12. Executable Content Tags
- 2.13. Summary
- 2.14. Exercise

---

### 2.0. OBJECTIVE

---

After going through this chapter you will be able to:

- Identify what is HTML
- Explain How and where to write HTML pages
- Explain use of Global Attributes
- Explain use of Event Handlers
- Identify different types of HTML tags, their functionality and attributes

---

### 2.1. WHAT IS HTML?

---

WebPages are written in HTML which is a simple client-side scripting language. HTML is short for HyperText Markup Language. **HyperText** is simply a piece of text that works as a link. **Markup Language** is a way of writing layout information within documents. Basically an HTML document is a plain text file that contains text and nothing else.

When a browser opens an HTML file, the browser will look for HTML codes in the text and use them to change the layout, insert images, or create links to other pages. Since HTML documents are just text files they can be written in even the simplest text editor.

A more popular choice is to use a special HTML editor, maybe one that puts focus on the visual result rather than the codes, a so called WYSIWYG editor ("What You See Is What You Get"). Some of the most popular HTML editors, such as FrontPage or Dreamweaver will let you create pages more or less as you write documents in Word or whatever text editor you are using. You can write your HTML by hand with almost any available text editor, including notepad that comes as a standard program with Windows.

All you need to do is type in the code, then save the document, making sure to put an **.html** extension or an **.htm** extension to the file (E.g., "mypage.html").

### HTML Tags

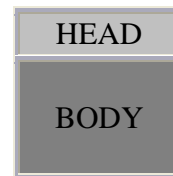
Everything is written inside HTML tags. A Tag should start with '`<`' and ends with '`>`'. HTML tags are mainly of two types:

- **Container Tag:** A Container tag is one that activates an effect and that has a *companion tag* that discontinues the effect. E.g., `<B>` is a container tag that, together with its companion closing tag `</B>`, causes all text found between them to be rendered in bold. The `<B>` tag turns on the bold effect and the `</B>` tag turns it off.
- **Standalone Tag:** A Standalone tag is one that does not have a companion tag. E.g., the `<IMG>` tag simply places an image on a page. `<IMG>` has no effect that was turned on and needs to be turned off, so no companion closing tag is needed.

### HTML Page Structure

All normal web-pages consist of a **HEAD** and a **BODY**.

- The **HEAD** is used for text and tags that do not show directly on the page.
- The **BODY** is used for text and tags that are shown directly on the page.



All web-pages have an `<HTML>` tag at the beginning and the end, telling the browser where the document starts and where it stops.

### Basic Code Structure

```

<HTML>
<HEAD>
<!-- This section is for the title and technical info of the page. -->
</HEAD>
<BODY>
<!-- This section is for all that you want to show on the page. -->

```

```
</BODY>
</HTML>
```

**The Tag's attributes:** An attribute modifies how a tag's effect is applied. Some tags take no attributes, and others may be able to take several.

---

## 2.2. GLOBAL ATTRIBUTES

---

Attributes that can be used with almost every tag are known as Global Attributes. Following is a list of global attributes:

Attribute	Value	Description
CLASS	classname	Specifies a classname for an element (used for stylesheets)
CONTENTEDITABLE	true false	Specifies if the user is allowed to edit the content or not
CONTEXTMENU	menu_id	Specifies the context menu for an element
DIR	ltr rtl	Specifies the text direction for the content in an element
DRAGGABLE	true false auto	Specifies whether or not a user is allowed to drag an element
DROPZONE	copy move link	Specifies what happens when dragged items/data is dropped in the element
HIDDEN	hidden	Specifies that the element is not relevant. Hidden elements are not displayed
ID	id	Specifies a unique id for an element
LANG	language_code	Specifies a language code for the content in an element fr" denotes French, "de" denotes German, etc.
SPELLCHECK	true false	Specifies if the element must have its spelling and grammar checked
STYLE	style definition	Specifies an inline style for an element
TABINDEX	number	Specifies the tab order of an element
TITLE	text	Specifies extra information about an element

**Let us see the progress:**

1. Lang attribute normally takes \_\_\_\_\_ character language code  
 a. 2                      b. 3                      c.1                      d.4
  
2. Which attribute takes values: true, false, auto?  
 a. spellcheck              b. draggable              c. contenteditable  
 d. dropzone
  
3. Which tag specifies the tab order of an element?  
 a. id                      b. tabindex              c. class                      d. style

**2.3. EVENT HANDLERS**

Within your HTML, you can respond to an event using an *event handler*. You can write an event handler to the HTML element for which you want to respond to when a specific event occurs. E.g., when a user double clicks on a thumbnail image you want to display full image. Here, the <IMG> tag which displays thumbnail image can have onDbClick event handler that will call script that will display full image.

**Events triggered for the window object apply to the <BODY> tag.**

<b>Event Handler</b>	<b>Description</b>
OnAfterPrint	Invokes the script after the document is printed
OnBeforePrint	Invokes the script before the document is printed
OnBeforeOnload	Invokes the script before the document loads
OnBlur	Invokes the script when the window loses focus
OnError	Invokes the script when an error occur
OnFocus	Invokes the script when the window gets focus
OnHasChange	Invokes the script when the document has change
Onload	Invokes the script when the document loads
OnMessage	Invokes the script when the message is triggered
OnOffline	Invokes the script when the document goes offline
OnOnline	Invokes the script when the document comes online

OnPageHide	Invokes the script when the window is hidden
OnPageShow	Invokes the script when the window becomes visible
OnPopState	Invokes the script when the window's history changes
OnRedo	Invokes the script when the document performs a redo
OnResize	Invokes the script when the window is resized
OnStorage	Invokes the script when a document loads
OnUndo	Invokes the script when a document performs an undo
OnUnload	Invokes the script when the user leaves the document

Events triggered by actions inside a HTML form apply to all HTML5 tags.

<b>Attribute</b>	<b>Description</b>
OnBlur	Invokes the script when an element loses focus
OnChange	Invokes the script when an element changes
OnContextMenu	Invokes the script when a context menu is triggered
OnFocus	Invokes the script when an element gets focus
OnFormChange	Invokes the script when a form changes
OnFormInput	Invokes the script when a form gets user input
OnInput	Invokes the script when an element gets user input
OnInvalid	Invokes the script when an element is invalid
OnSelect	Invokes the script when an element is selected
OnSubmit	Invokes the script when a form is submitted



**Events triggered by a keyboard apply to all HTML5 tags.**

<b>Attribute</b>	<b>Description</b>
OnKeyDown	Invokes the script when a key is pressed
OnKeyPress	Invokes the script when a key is pressed and released
OnKeyUp	Invokes the script when a key is released

**Events triggered by a mouse apply to all HTML5 tags.**

<b>Attribute</b>	<b>Description</b>
OnClick	Invokes the script on a mouse click
OnDbClick	Invokes the script on a mouse double-click
OnDrag	Invokes the script when an element is dragged
OnDragEnd	Invokes the script at the end of a drag operation
OnDragEnter	Invokes the script when an element has been dragged to a valid drop target
OnDragLeave	Invokes the script when an element leaves a valid drop target
OnDragOver	Invokes the script when an element is being dragged over a valid drop target
OnDragStart	Invokes the script at the start of a drag operation
OnDrop	Invokes the script when dragged element is being dropped
OnMouseDown	Invokes the script when a mouse button is pressed
OnMouseMove	Invokes the script when the mouse pointer moves
OnMouseOut	Invokes the script when the mouse pointer moves out of an element
OnMouseOver	Invokes the script when the mouse pointer moves over an element

OnMouseUp	Invokes the script when a mouse button is released
OnMouseWheel	Invokes the script when the mouse wheel is being rotated
OnScroll	Invokes the script when an element's scrollbar is being scrolled

---

## 2.4. DOCUMENT STRUCTURE TAGS

---

The document structure tags are those that define each component. Every HTML document has three major components:

- the HTML declaration
- the HEAD, and
- the BODY

Following is a list of document structure tags:

Tag	Example	Description
<HTML>	<HTML> ... all content and HTML code goes here... </HTML>	Declares the document to be an HTML document
<HEAD>	<HTML> <HEAD> ... </HEAD> ... </HTML>	Contains the tags that compose the document head
<BASE>	<HEAD> <BASE HREF="http://www.myserver.com/xml/" TARGET="_blank"/> </HEAD> <BODY> <A HREF="default.asp">XML file</A> </BODY>  This example sets XML as default directory so all files under this directory and can be accessed by specifying only name of the file rather than giving the whole path for the file	The <BASE> tag specifies a default URL, and / or a default target, for all elements with a URL <b>Attributes:</b> HREF - The URL to use as the base URL for links in the page TARGET - Specifies where to open all the links on the page. Can take values: _blank, _parent, _self, _top, or framename

<META>	<p>&lt;META NAME="keywords" CONTENT="HTML, XML, JavaScript,PHP"/&gt; This example define keywords for search engines</p> <p>&lt;META HTTP-EQUIV="refresh" CONTENT="5"/&gt; This example will refresh page every 5 seconds</p>	<p>META elements are typically used to specify page description, keywords, author of the document, last modified and other metadata. These elements are used by browsers or search engine.</p> <p><b>Attributes:</b>  <b>CHARSET</b> - Specifies the character encoding for the document  <b>CONTENT</b> - Value of meta variable  <b>HTTP-EQUIV</b> - Specifies a HTTP header for the information in the content attribute. Can take values: expires, refresh  <b>NAME</b> - Specifies a name for the meta variable. Can take values: author, description, keywords, generator</p>
<LINK>	<p>&lt;LINK HREF="/style/styles.css" REL="Stylesheet"&gt; &lt;LINK HREF="/index.html" REL="home"&gt;</p>	<p>Defines the relationship between a document and a linked document.</p> <p><b>Attributes:</b>  <b>HREF</b> – Set equal to the URL of the linked document  <b>HREFLANG</b> - A two-letter language code that specifies the</p>

		<p>language of the linked document. E.g., 'en'</p> <p><b>MEDIA</b> - Specifies what media / device the target URL is best. E.g., printer, speaker, monitor, etc.</p> <p><b>REL</b> - Specifies the relationship between target document and current document. E.g., stylesheet, home, help, toc, glossary, etc.</p> <p><b>TYPE</b> - Specifies the MIME type of the target URL. E.g., text/css, text/javascript, image/gif</p>
<p>&lt;SCRIPT&gt;</p>	<pre>&lt;SCRIPT TYPE="text/javascript"&gt; document.write ("Hello World!") &lt;/SCRIPT&gt; &lt;SCRIPT TYPE="text/javascript" SRC="first.js"&gt; &lt;/SCRIPT&gt;</pre>	<p>Used to define a client-side script. The script element either contains scripting statements or it points to an external script file through the SRC attribute.</p> <p><b>Attributes:</b></p> <p><b>TYPE</b> - Specifies MIME type of script. The default value is "text/javascript"</p> <p><b>CHARSET</b> - Defines the character encoding used in script</p> <p><b>DEFER</b> - Indicates that the script is not going to generate any</p>

		document content. The browser can continue parsing and drawing the page SRC - Defines a URL to a file that contains the script
<STYLE>	<HTML> <HEAD> <STYLE TYPE="text/css"> H1 {color:yellow} P {color:blue} </STYLE> </HEAD> <BODY> <H1>Header 1</H1> <P>A paragraph.</P> </BODY> </HTML>	Specifies style information for the document. <b>Attributes:</b> MEDIA - Specifies what media / device the target URL is best. E.g., printer, speaker, monitor, etc. TYPE - Specifies the MIME type of the style sheet.
<TITLE>	<TITLE>My First Web page</TITLE>	Gives a descriptive title to a document. Defines a title in the browser toolbar.
<BDO>	<BDO DIR="LTR">Here's some English text.</BDO>	Allows you to specify the text direction and override the bidirectional algorithm. <b>Attribute:</b> DIR - specifies direction to read, left-to-right, right-to-left
<BODY>	<BODY> Text and tags related to document </BODY>	Contains all content and tags that compose the document body <b>Attributes:</b> BGCOLOR- background color BACKGROUND- background

		image LINK- unvisited link color ALINK- active link color VLINK- visited link color TEXT- text color Note: All attributes are removed in HTML5
--	--	---

**Note:**

Both style and script tags are normally written inside comment (<!-- ... -->). So the browsers that do not support script or style can ignore this tag's content. And browsers that support style and script can ignore comment and parse the tag's content

**Let us see the progress:**

4. What value type attribute of <link> tag will take when the linked document is a stylesheet file  
 a. text/css      b. text/stylesheet      c. text/ss      d. text/cs
5. How many values generally taken by http-equiv attribute?  
 a. 2              b. 1              c. 3              d. 4
6. If I want to write content in English and Urdu in my html page. Which tag will be useful to specify reading direction to the browser?  
 a. dir              b. bdo              c. body              d. html

---

## 2.5. FORMATTING TAGS

---

As the name suggests, these tags can be used to format or change the view or display of the text that is shown on the web page. There are 2 types:

**1. Text Formatting Tags****a. Font Formatting Tags: Changes the font properties of the text.**

Tag	Example	Description	Output
<B>	<B>text</B>	Writes text as bold	<b>text</b>
<BASEFONT>	<BASEFONT SIZE=5 COLOR='gray' FACE='Arial, verdana, Times'>	Sets base size, color, and typeface properties for the body text font	

<BIG>	Big<big>text</big>	Increase the size by one	<b>Big text</b>
<FONT>	<FONT SIZE="1" COLOR=gray FACE="Arial"> text </FONT>  <FONT SIZE="7"> text </FONT>	Writes text in smallest font size. (8 pt)  Writes text in biggest font size (36 pt)	<small>text</small>  <b>text</b>
<I>	<I>text</I>	Writes text in italics	<i>Text</i>
<S> <STRIKE>	<S> text </S> <Strike> text </Strike>	Strikes a line through the text	<del>Text</del>
<SMALL>	Small <SMALL>text</SMALL>	Decrease the size by one	<small>Small text</small>
<SUB>	<SUB> text </SUB>	Lowers text and makes it smaller	<sub>text</sub>
<SUP>	<SUP> text </SUP>	Lifts text and makes it smaller	<sup>text</sup>
<TT>	<TT> text </TT>	Writes text as on a classic typewriter	<b>Text</b>
<U>	<U> text </U>	Writes underlined text	<u>Text</u>

**b. Phrase Formatting Tags:** Describe how the text is being used in the context of the document. It tells us what the content of tag is.

Tag	Example of tag	Description of Tag
<ABBR>	She got her Doctorate (<ABBR>PhD</ABBR>) from the University of Mumbai	Contains text that is an abbreviation of something. Useful for speech based browser.
<ACRONYM>	Practical Extraction and Reporting Language <ACRONYM>(PERL)</ACRONYM> is a popular CGI Scripting Language	Contains text that specifies an acronym. Useful for speech based browser.
<ADDRESS>	If you have any comments, please send them to <ADDRESS>admin@mu.co.in</ADDRESS>.	Contains either a postal or an electronic email address, typically rendered in italics.


<CITE>	According to the <CITE>HTML 4.0 Recommendation</CITE>, the <FONT> tag has been deprecated	Contains the name of a source from which a passage is referred, typically rendered in italics.
<CODE>	<CODE> LOCATION.HREF ='index.html'; Return true; </CODE>	Contains chunks of computer language code, displays in 'Courier' font face.
<DEL>	John just got a big<DEL>, huge</DEL> basket	Contains text that has been deleted from the document by editor. <b>Attributes:</b> CITE - Provides the URL of a document that explains why the deletion was necessary DATETIME - Puts a "timestamp" on the deletion
<DFN>	HTML uses the notion of <DFN>client pull</DFN> - a dynamic document technique	Denotes the defining instance of a term. Internet Explorer will display the text inside <DFN> in italics.
<EM>	Please do <EM>not</EM> disturb the guard	Contains text to be emphasized. Most browsers render emphasized text in italics.
<INS>	The New World was discovered by <DEL>Peter</DEL> <INS>Columbus</INS> in 1492	Contains text that has been inserted into the document by editor. <b>Attributes:</b> CITE - Provides the URL of a document that explains why the insertion was necessary DATETIME - Puts a "timestamp" on the insertion
<KBD>	To begin, type <KBD>go</KBD> and press Enter	Contains text that represents keyboard input.



<Q>	<Q CITE="www.education.org"> Education is must</Q>	Contains a direct quotation to be displayed inline. CITE will have URL of site from which quotation is taken.
<SAMP>	A common first exercise in a programming course is to write a program to produce the message <SAMP>Hello World</SAMP>	Contains text that represents the exact output from a program.
<STRONG>	<STRONG>STOP! </STRONG> Do not proceed any further.	Contains text to be strongly emphasized.
<VAR>	The <VAR>Result</VAR> variable is set to the number of records that the query retrieved	Denotes a variable from a computer program.

**2. Block Formatting Tags:** This type of tag formats a specific logical part of the document.

Tag	Example	Description	Output
<BLOCKQUOTE>	India is a <BLOCKQUOTE> Democratic </BLOCKQUOTE> Country	Contains quoted text is to be displayed indented from body text CITE will have URL of the site from which quotation is taken	India is a  Democratic  Country
 	There are two types of schools: Public and  Private	Inserts a line break in the document. Carriage returns in the HTML code do not output to line breaks on the browser screen, so one often need to insert the breaks themselves	There are two types of schools: Public and Private

<DIV>	<DIV ALIGN="center"> text1</DIV> <DIV ALIGN="left"> text2</DIV>	Defines a section or division of a document that requires a special alignment	text1 text2
<HR>	<HR NOSHADE WIDTH=80% SIZE=4>	Places a horizontal line on the page. <b>Attributes:</b> ALIGN - Left, Right, or Center NOSHADE - Removes the shading effect and yields a solid line SIZE - Controls the thickness of line WIDTH - Length of the line. Can be specified in pixels or %	
<H1> <H2> <H3> <H4> <H5> <H6>	<H1> Table of Contents </H1> <H2> Chapter 1 – Introduction </H2> <H3> Objectives </H3> <H4> Sub objectives </H4>	Displays document headings levels. Level1 has the largest font size. <H1> will display text in largest size. <H6> will display text in smallest size. As level increases the heading size decreases	<b>Table of Contents</b> Chapter 1 – Introduction Objectives Sub objectives
<P>	Paragraph starts here. <P>some text that will show us how rendering is done by paragraph tag</P> Paragraph ends here.	Defines a paragraph. It automatically creates some space before and after itself. Takes more vertical space.	Paragraph starts here. some text that will show us how rendering is done by paragraph tag Paragraph ends here.

<PRE>	<pre> &lt;pre&gt; Name Age Gender Amit 19 Male Priya 18 Female &lt;/pre&gt; </pre>	Defines preformatted text. Text in a pre element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.	<table> <tr> <td>Name</td> <td>Age</td> <td>Gen</td> </tr> <tr> <td>Amit</td> <td>19</td> <td>Mal</td> </tr> <tr> <td>Priya</td> <td>18</td> <td>Fem</td> </tr> </table>	Name	Age	Gen	Amit	19	Mal	Priya	18	Fem
Name	Age	Gen										
Amit	19	Mal										
Priya	18	Fem										
<SPAN>	<pre> &lt;SPAN STYLE= "font-weight: bold; color: gray"&gt; Here is some bold, red, text &lt;/SPAN&gt; </pre>	Used to group inline-elements in a document. One popular use is for applying style information.	<b>Here is some bold, red</b>									

**Let us see the progress:**

7. Which tags have size, color, and face attributes?
  - a. Basefont
  - b. font
  - c. hr
  - d. div
  
8. I have done formatting of a letter in Microsoft word. Now I want that letter to be included in HTML document. Which tag I can use?
  - a. Span
  - b. Div
  - c. Pre
  - d. P
  
9. An author has written an article on global warming. Editor1 has inserted some text and deleted some text. Editor2 should also be able to view the changes done by Editor1. How is it possible to do this in HTML/on line?
  - a. samp and var tag
  - b. del and ins tag
  - c. sub and sup tag
  - d. I and S tag

---

## 2.6. LIST TAGS

---

Lists are a useful way of representing content.

Tag	Example	Description	Output
<LI>	<pre> &lt;LI&gt;Red&lt;/LI&gt; &lt;LI&gt;Blue&lt;/LI&gt; &lt;LI&gt;Yellow&lt;/LI&gt; </pre>	Denotes an item in a list. The <LI> tag is always used in conjunction with <OL>, and <UL> tags	<ul style="list-style-type: none"> <li>• Red</li> <li>• Blue</li> <li>• Yellow</li> </ul>
<OL>	<pre> &lt;OL TYPE="A" Start=3&gt; &lt;LI&gt;HTML&lt;/LI&gt; &lt;LI&gt;XML&lt;/LI&gt; &lt;LI&gt;PHP&lt;/LI&gt; &lt;LI&gt;JavaScript&lt;/Li&gt; </pre>	Creates an ordered or numbered list. <p><b>Attributes:</b></p> START - Changes to a position other than the ordering scheme. E.g., setting START to 3 with TYPE set equal to I produces a list that begins	<ol style="list-style-type: none"> <li>3. HTML</li> <li>4. XML</li> <li>5. PHP</li> <li>6. JavaScript</li> </ol>

	</OL>	numbering with III (3 in uppercase roman numerals). TYPE - Controls the numbering scheme. Can take one of the value "1 A a I i" A - Uppercase Letters a - Lowercase Letters I - Uppercase Roman Numerals i - Lowercase Roman Numerals Note: TYPE is not supported in HTML5	
<UL>	<UL TYPE="disc"> <LI>HTML</LI> <LI>XML</LI> <LI>PHP</LI> <LI>JavaScript</LI> </UL>	Creates an unordered or bulleted list. <b>Attributes:</b> TYPE - Specifies which bullet character to use when displaying the list. Can take one of the value "DISC SQUARE CIRCLE" <ul style="list-style-type: none"> <li>• DISC - solid circular bullet</li> <li>▪ SQUARE - solid square bullet</li> <li>○ CIRCLE- open circular bullet</li> </ul> Note: TYPE is not supported in HTML5	<ul style="list-style-type: none"> <li>• HTML</li> <li>• XML</li> <li>• PHP</li> <li>• JavaScript</li> </ul>
<DL>	<DL> <DT>Telnet</DT> <DD>For remote login</DD> <DT>FTP</DT> <DD>For file transfer</DD> </DL>	Denotes a definition list	Telnet For remo FTP For file tr

**Let us see the progress:**

10. Which tags will be used to create glossary of a book?

- a. UL, LI      b. DL, DT, DD      c. OL, LI      d. OL,UL,LI

11. In which tag type attribute takes five values?

- a. DL      b. UL      c. OL      d. DT

---

## 2.7. HYPERLINK TAGS

---

The <A> tag defines a hyperlink, which is used to link from one page to another. By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

Attribute	Value	Description
HREF	URL or anchor name	The URL of the link. Possible values: <ul style="list-style-type: none"> <li>• Absolute URL - Points to another website. E.g., HREF="http://www.myserver.com/index.htm"</li> <li>• Relative URL - Points to a file within a website. E.g., HREF="index.htm"</li> <li>• Anchor URL - Points to an anchor within a page. E.g., HREF="#top"</li> </ul>
HREFLANG	Language code	A two-letter language code that specifies the language of the linked document. E.g., 'en'
MEDIA	Media query	Specifies what media/device the target URL is best. E.g., printer, speaker, monitor, etc.
NAME	Name of anchor	Specifies the name of the anchor being set up
REL	Alternate, archives, help, author, first, last, index, license, next, search, stylesheet, etc.	Specifies the relationship between target document and current document
TARGET	_blank _parent _self _top <i>framename</i>	Tells the browser into which frame the linked document should be loaded
TYPE	Audio, video, image, text, etc.	Specifies the MIME type of the linked resource

Following code illustrates hyperlink:

```
<A HREF="prod.html" TARGET="_blank">Products Detail Page</A>
```

To follow the link, a user can click the hypertext 'Products Detail Page'

Following code establishes a named anchor within a document:

```
//Placing anchor tag
<A NAME="top">
<H1>Table of Contents</H1>
</A>
... // some HTML tags and content
// at the end of page
<A HREF="#top">TOP</A>
```

When a user is at the end of page, he can click on 'Top' to reach to the place where anchor tag is placed i.e. on the top of the page.

Suppose if anchor tag is anchor page then use following:

```
<A HREF="product.htm#top">TOP</A>
```

Let us see the progress:

12. Target attribute of <a> tag cannot take following value?

- a. \_parent   b. \_blank   c. \_frameName   d. \_self

13. Which two attributes are must to navigate within page?

- a. type   b. href   c. name   d. hreflang

---

## 2.8. IMAGE AND IMAGEMAPS

---

To place an image on a HTML page <IMG> tag is used. To create a multi-link image <MAP> and <AREA> tags are used along with <IMG> tag.

Tag	Example	Description
<IMG>	<IMG WIDTH=600 HEIGHT=120 SRC= "/images/logo.gif" ALT="Welcome to XYZ Corporation" USEMAP="#main">	Places an inline image into a document. Pictures, logos, and other graphical effects are placed into a document using the <IMG> tag.  <b>Attributes:</b> SRC - Specifies the URL of the image file ALT - A text-based description of the image content (alternative text). Many browser display it as tooltip when mouse is moved over the image WIDTH and HEIGHT - Gives the width and height of the image in pixels

		<p>ISMAP - Identifies the image as being used as a part of a server-side imagemap</p> <p>USEMAP - Set equal to the name of the client-side imagemap to be used with the image</p>
<MAP>	<pre>&lt;MAP NAME="main"&gt; &lt;AREA SHAPE="POLY" HREF="profile.html " COORDS="35,80,16 8,99,92,145"&gt;</pre>	<p>Contains HTML tags that define the clickable regions (hot regions) of an imagemap.</p> <p><b>Attribute:</b></p> <p>NAME - Gives the map information a unique name so it can be referenced by the USEMAP attribute in the &lt;IMG&gt; tag that places the imagemap graphics</p>
<AREA>	<pre>&lt;AREA SHAPE="CIRCLE" HREF="feedback.ht ml" COORDS="288,306, 288,334"&gt; &lt;AREA SHAPE="DEFAULT" HREF="index.html" &gt; &lt;/MAP&gt;</pre>	<p>Defines a hot region in a client-side imagemap.</p> <p><b>Attributes:</b></p> <p>ALT - Provides a text alternative for the hot region in the event that the image does not load</p> <p>SHAPE - Specifies the shape of the hot region being defined. Possible values of SHAPE include:</p> <ul style="list-style-type: none"> <li>RECT - rectangles</li> <li>CIRCLE - circles</li> <li>POLY - polygons</li> <li>DEFAULT - for any point on the image not part of another hot region</li> </ul> <p>HREF - Set equal to the URL of the document to associate with the hot region</p> <p>HREFLANG - A two-letter language code that specifies the language of the linked document. E.g., 'en'</p> <p>MEDIA - Specifies what media/device the target URL is best. E.g., Printer, speaker, monitor, etc.</p> <p>REL - Specifies the relationship between target document and current document</p> <p>TARGET - Specifies into which frame to load the linked document</p>

		<p>TYPE - Specifies the MIME type of the target URL</p> <p>COORDS - Specifies the coordinates that define the hot region</p>								
		<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><math>x1, y1, x2, y2</math></td> <td>If SHAPE="rect", then coordinates of the top-left corner and the bottom-right corner of the rectangle</td> </tr> <tr> <td><math>x, y, radius</math></td> <td>If SHAPE="circle", then coordinates of the circle center and the radius</td> </tr> <tr> <td><math>x1, y1, x2, y2, \dots, xn, yn</math></td> <td>If SHAPE="poly", then coordinates of the edges of the polygon. If the first and last coordinate pairs are the same, the browser will add a third coordinate pair to close the polygon</td> </tr> </tbody> </table>	Value	Description	$x1, y1, x2, y2$	If SHAPE="rect", then coordinates of the top-left corner and the bottom-right corner of the rectangle	$x, y, radius$	If SHAPE="circle", then coordinates of the circle center and the radius	$x1, y1, x2, y2, \dots, xn, yn$	If SHAPE="poly", then coordinates of the edges of the polygon. If the first and last coordinate pairs are the same, the browser will add a third coordinate pair to close the polygon
Value	Description									
$x1, y1, x2, y2$	If SHAPE="rect", then coordinates of the top-left corner and the bottom-right corner of the rectangle									
$x, y, radius$	If SHAPE="circle", then coordinates of the circle center and the radius									
$x1, y1, x2, y2, \dots, xn, yn$	If SHAPE="poly", then coordinates of the edges of the polygon. If the first and last coordinate pairs are the same, the browser will add a third coordinate pair to close the polygon									

### Let us see the progress:

14. If in <AREA> tag shape is taken as poly then which coordinates are supposed to be considered?
- a. top-left corner, bottom-right corner      b. top-right corner, bottom-left corner
- c. edges of the geometric shape              d. center, radius
15. Which attribute will let us know whether image is part of server side scripting?
- a. usemap              b. ismap              c. isusemap              d. src
16. Value of which attribute of <MAP> tag is referenced in <IMG> tag to link the graphic and imagemap information?
- a. shape              b. name              c. area              d. coords

---

## 2.9. TABLE TAGS

---

HTML table tags are better way of representing information in a web page. These give systematic representation of data. Following are the HTML table tags:

<TABLE>, <CAPTION>, <THEAD>, <TFOOT>, <TBODY>, <COLGROUP>, <COL>, <TR>, <TD>, and <TH>

For detailed explanation of HTML table tags refer to Chapter 4.



---

## 2.10. FORM TAGS

---

HTML forms are allowing web user to interact with the website. Forms collect information from a user, and then a script or program on a web server uses this information to compose a custom response to the form submission. These form tags are used to produce the form and form controls. Following are the HTML form tags:

<FORM>, <INPUT>, <SELECT>, <OPTION>, <OPTGROUP>, <TEXTAREA>, <BUTTON>, <LABEL>, <FIELDSET>, and <LEGEND>

For detailed explanation of HTML table tags refer to Chapter 6.

---

## 2.11. FRAME TAGS

---

Framed Layouts allows the browser window to split into multiple regions called frames. Each frame can contain a distinct HTML document, enabling you to display several documents at once, rather than just one. Frame tags enable you to keep key page elements (such as navigation links) on the screen all the times, while other parts of the page change. Following are the HTML frame tags:

<FRAMESET>, <FRAME>, <NOFRAME>, and <IFRAME>

**Note:** Except <IFRAME> all other tags are not supported in HTML5

For detailed explanation of HTML table tags refer to Chapter 5.

---

## 2.12. EXECUTABLE CONTENT TAGS

---

Web pages have become more dynamic is due to executable content, such as Java applets and ActiveX controls. These page elements are downloaded to the browser and run in its memory space to produce dynamic content on the browser screen. Following are the HTML executable content tags:

<APPLET>, <PARAM>, and <OBJECT>

Note: Except <Object> all other tags are not supported by HTML5

Tag	Example	Description
<APPLET>	<APPLET WIDTH=250 HEIGHT=200 CODE="effect.class" NAME="effect" ALT="effect text applet" HSPACE=5 VSPACE=12	Places a JAVA Applet on a page <b>Attributes:</b> CODE - Specifies class file name ALIGN - top   middle   bottom   left   right

	<p>ALIGN="right"&gt;          &lt;PARAM          NAME="message"          VALUE="Be Effective!"&gt;          ...          &lt;/APPLET&gt;</p>	<p>ALT - Alternative text          ARCHIVE - Archive list          CODEBASE - URL for applet code          HEIGHT and WIDTH - Specifies the width &amp; height in pixels          HSPACE and VSPACE - Controls the amount of white space around the applet (in pixels)          NAME - applet name          OBJECT - Name of serialized applet file</p>
<PARAM>		<p>Passes a parameter to a JAVA Applet.  <b>Attributes:</b>          ID - Unique identifier          NAME - Parameter name          VALUE - Parameter value          VALUETYPE - data   ref   object          TYPE - Internet media type</p>
<OBJECT>	<p>&lt;OBJECT          DATA="inlineframe.swf"&gt;          &lt;/OBJECT&gt;</p>	<p>Used to include objects such as images, audio, videos, JAVA Applets, ActiveX, PDF, and Flash  <b>Attributes:</b>          DATA - URL for the data of the object          FORM - Specifies one or more forms the object belongs to          WIDTH and HEIGHT - Specifies the width &amp; height of the object in pixels          Name - Unique name of the object          TYPE - MIME type of data          USEMAP - Name of the &lt;MAP&gt; tag for client-side imagemapping</p>

**Let us see the progress:**

17. Which attribute takes name of the class file of the applet code to place an applet on the HTML page?  
 a. code                      b. codebase                      c.name                      d. object
18. Which tag is used to include objects such as images, audio, videos, JAVA Applets, ActiveX, PDF, and Flash in HTML page?  
 a. applet                      b. object                      c. param                      d. appobj

**2.13. SUMMARY**

In this chapter we have learnt how to create HTML pages, different event handlers, what is use of global attributes. We have discussed various types of HTML tags, attributes and their uses. Thus we now know what can go inside an html page and how to create better web pages.

Answers of check your progress:

- |       |         |          |       |       |
|-------|---------|----------|-------|-------|
| 1. a  | 2. b    | 3. b     | 4. a  | 5. a  |
| 6. b  | 7. a, b | 8. c     | 9. b  | 10. b |
| 11. c | 12. c   | 13. b, c | 14. C |       |
| 15. b | 16. b   | 17. b    | 18. b |       |

**2.14. EXERCISE****2.14.1. Questions**

1. What is HTML? Why we require HTML?
2. How to create an HTML document?
3. Define tag's attribute? What are global attributes? Explain them in detail.
4. What are event handlers? Explain the event handlers that can be used with body tag.
5. Explain event handlers that can be used with mouse movements done by user.
6. Explain <BASE>, <ISINDEX>, <META>, <LINK>, <BODY> document structure tags.
7. Explain text level font formatting tags.
8. Explain text level phrase formatting tags.
9. Explain block level formatting tags.
10. Explain how to place ordered and unordered list on web page. Explain related tags in detail.
11. How to place hyperlink on web page? Explain <A> tag in detail.
12. Explain tags that are useful for placing image and imagemap on web page.
13. What is role of Executive content tags? Explain <object> tag in detail.

**2.14.2. Programs**

1. Write code for an HTML page that will use font formatting tags to display information.
2. Write code for an HTML page that will use block level formatting tags to display information.
3. Write code for the following output using list tags:
  1. F.Y.B.Sc.(IT)
    - i. New Syllabus
    - ii. New Question format
  2. S.Y.B.Sc.(IT)
    - i. Old Syllabus
    - ii. Old Question format



## IMAGEMAPS AND STYLESHEETS

### Unit Structure

- 3.0. Objective
- 3.1. What are Imagemaps?
- 3.2. Client-Side Imagemap
- 3.3. Server-Side Imagemap
- 3.4. Using Server-Side and Client-Side Imagemap Together
- 3.5. Alternative Text for Imagemap
- 3.6. What are Style Sheets?
- 3.7. Why are Sheets Valuable?
- 3.8. Different Approaches to Style Sheets
- 3.9. Using Multiple Approaches
- 3.10. Linking to Style Information in Separate File Using <LINK> Tag
- 3.11. Setting up Style Information
- 3.12. Embedded Style Information Using <STYLE> Tag
- 3.13. Inline Style Information
- 3.14. Summary
- 3.15. Exercise

---

### **3.0. OBJECTIVE**

---

After going through this chapter you will be able to:

- Identify what are imagemaps
- Create client-side and server-side imagemaps
- Using client-side and server-side imagemaps together
- Providing alternative text for imagemaps
- Explain the role of Style sheets
- Identify how to include style sheet in your web page i.e. different approaches to style sheets

---

### 3.1. WHAT ARE IMAGEMAPS?

---

Imagemaps are images with clickable areas (sometimes referred to as "hotspots") that usually link to another page. This image is "multi-linked" and can take you to a number of places (pages). Such a multilinked image is called an **Imagemap**. If used tactfully, imagemaps can be really effective. If not, they can potentially confuse users.

The important task in preparing an imagemap is defining which parts of the image are linked to which URLs. Linked regions in an imagemap are called *hot regions*, and each hot region is associated with the URL of the document that is to be loaded when the hot region is clicked.

#### **Pencil Sharpeners Results**



**There are two types of imagemaps:**

1. Client-Side Imagemap

In this type of Imagemapping, information about the hot region is downloaded on the client machine with HTML page. So any time (offline) client can use this Imagemap information. This method is more preferred.

2. Server-Side Imagemap

In this type of Imagemapping, information about the hot region is lying on the server. When any area or hot region of the Imagemap graphic is clicked, coordinates of that area are passed by browser to the server. Then server does processing and sends the output.

---

## 3.2. CLIENT-SIDE IMAGEMAP

---

Client-side imagemaps gives faster imagemap processing and enhance the portability of your HTML documents. Client-side imagemaps involve sending the map data to the client as part of an HTML file rather than having the client contact the server each time the map data is needed.

### Advantages

1. **Supported by HTML and the Browser:** The main advantage for client-side image maps is that all the features of the image map (hotspots, related hyperlinks and image) can all be supported with HTML and rendered by the browser.
2. **Server is not involved:** Client-side image maps can be designed and implemented without regard to any special server process requirements.
3. **Faster Design:** Because the design of the image map has been simplified by eliminating the server process, you are able to build your image map faster.
4. **Faster Links:** Because coordinates and destination links are resolved on the client (as opposed to extra traffic to and from the server), the links are resolved faster.

### Steps for creating a client-side imagemap

There are three steps involved in creating a client-side imagemap. They are as follows:

1. Create the graphic (image) that you want to map into an imagemap.
2. Define the hot regions for the graphic (using <AREA> tag) and place that information between the <MAP> and </MAP> tags in your HTML document.
3. Use the <IMG> tag to insert the graphic for the imagemap and link it to the hot region information you defined in the <MAP>section by using USEMAP attribute of <IMG> tag.

To set up a circular hot region, you would use code such as the following:

```
<MAP NAME="circle">  
<AREA SHAPE="circle" COORDS="123, 89, 49" ALT="Circle  
Link" HREF=http://www.myserver.com/circle.html>  
</MAP>
```

Now this mapping information should be linked with an image using:

```
<IMG SRC="images/circular.gif" USEMAP="#circle">
```

The pound sign (#) before the map name indicates that the map data is found in the same HTML file. If the map data is in another file called `imagemap.html`, your `<IMG>` tag would look like the following:

```
<IMG SRC="images/circular.gif"
USEMAP="http://www.myserver.com/imagemap.html#circle">
```

E.g., creating client-side mapping for the following image

As we can see there are three shapes viz., circle, rectangle and triangle, in the image below which is saved as `shapes.gif`.



First Create Hot regions:

```
<MAP NAME=shape>
<AREA SHAPE="circle" COORDS="23,24,20"
HREF=circle.html ALT="circle clicked">
<AREA SHAPE="rect" COORDS="28,61,92,93"
HREF=rect.html ALT="rectangle clicked">
<AREA SHAPE="poly" COORDS="80,4,63,37,98,33,80,4"
HREF=poly.html ALT="triangle clicked">
</MAP>
```

Link hot region with image:

```
<IMG SRC="shapes.gif" ALT="showing different shapes"
USEMAP=#shape>
```

Note: If `<MAP>` tag is in some other file (`work.htm`) then use:

```
<IMG SRC="shapes.gif" ALT="showing different shapes"
USEMAP=work.htm#shape>
```

---

### 3.3. SERVER-SIDE IMAGEMAP

---

A server-side imagemap is one in which the server determines which document should be loaded, based on where the user clicked the imagemap. For this purpose, the server needs the following information:

1. The coordinates of the user's click - When user will click on any part of the image, coordinates of that point will be sent to the server. This information is passed to the server by the client program.
2. A program that takes the click coordinates as input and provides a URL as output - Most of the server has their own mechanism to do this task.



3. Access to the information that defines the hot regions and their associated URLs - Information about hot region and linked URL in a file stored on server called map file. This file information is critical to the processing program to a URL. When the program finds a match, it returns the URL paired with the clicked hot region.
4. An image/graphic - That is supposed to be used as Imagemap.
5. Proper setup in your HTML file - When you place the imagemap graphic, you use the <IMG> tag with 'ISMAP' attribute to alert the browser that the image is to be used as a server-side imagemap.

### **Creating Map File**

The map file is a text file that contains information about the hot regions of a specific imagemap graphics. A separate map file is needed for each imagemap graphic.

### **Points to remember:**

1. If two regions overlap in their coordinates, the imagemap program uses the first region it encounters in the file
2. URLs in map files should always be absolute or fully qualified URLs
3. Pound sign (#) can be used to insert comment on a line in the map file
4. If shape is circle then coordinates of center and coordinates of any point on circumference is to be specified

Format of the map file according to NCSA (National Center for Supercomputing Applications) is: region\_type URL coordinates

E.g.,

circle <http://www.myserver.com/circle.html> 123, 89, 46,132

### **Steps for creating a server-side imagemap**

There are three steps involved in creating a client-side imagemap. They are as follows:

1. Create the graphic (image) that you want to map into an imagemap.
2. Create map file
3. Link map file with image by using 'ISMAP' attribute of <IMG> tag and putting <IMG> tag into a hyperlink where 'HREF' is pointing to map file

E.g., creating server-side imagemap for the following image

As we can see there are three shapes viz., circle, rectangle and triangle, in the image below which is saved as shapes.gif.



First create a map file that will contain information about hot region. Type the following in notepad and save as shapes\_map.map:

```
circle http://myserver.com/map/circle.html 23,24,44,21
rect http://myserver.com/map/rect.html 28,61,92,93
poly http://myserver.com/map/poly.html
80,4,63,37,98,33,80,4
```

Setting ISMAP attribute and using <A> tag:

```
<A HREF= http://myserver.com/map/shapes_map.map>
  <IMG SRC=shapes.gif ISMAP>
</A>
```

When the link is clicked, the browser will request the given link, and add “?x,y” at the end of it, as the click offset from the left, top corner of the image (such as Shapes\_map.map?47,8). If the user is not using a mouse (or equivalent), then the coordinates will be 0,0.

**Let us see the progress:**

1. Which attribute of <img> tag doesn't take any value and is useful for server side imagemaps?
  - a. usemap
  - b. ismap
  - c. src
  - d. alt
2. Which coordinates are required for shape=circle if we are using server side Imagemap?
  - a. center, any point on circle
  - b. center, radius
  - c. radius, any point on circle
  - d. center, diameter

---

### 3.4. USING SERVER-SIDE AND CLIENT-SIDE IMAGEMAP TOGETHER

---

Client-side imagemaps gives faster imagemap processing and enhance the portability of your HTML documents. Server-side imagemaps hides mapping information. You can combine server-side and client-side imagemaps, implementing both at the same time, to ensure your imagemaps are accessible to the broadest possible audience.

To combine a server-side imagemap with a client-side imagemap for the shapes.gif example discussed earlier, you can modify the earlier HTML as follows:

```
<A HREF= http://myserver.com/map/shapes_map.map>
<IMG SRC="shapes.gif" USEMAP=#shape ISMAP>
</A>
```

Putting the <IMG> tag with <A> and </A> tags makes it point to the shapes\_map.map file on the server. You need to include the ISMAP attribute in the <IMG> tag to let the browser know that the image is linked as a server-side imagemap as well.

NOTE - You can link NCSA and CERN style server-side imagemaps to client-side imagemaps by having the HREF in the <A> tag point to the imagemap script instead of pointing directly to the map file.

---

### 3.5. ALTERNATIVE TEXT FOR IMAGEMAP

---

When you use a server-side imagemap, it is important to provide a text-based alternative to users who have a text-only browser, who have image loading turned off, or who are using a non-visual browser. These users will not be able to view your image, so the entire imagemap will be lost on them if a text-based alternative is not supplied.

To do this we have to insert links to the mapped pages at the end of the imagemap html page. And in the imagemap graphic i.e. in <IMG> tag give ALT="imagemap links are at the bottom of the page".

Text-based alternative are less critical for client-side imagemaps because of the ALT attribute of the <AREA> tag.

Most sites place their text-based alternative to an imagemap just below the imagemap graphic. Usually the links are in a smaller font size and are separated by vertical bars or some such separator character.

**Let us see the progress:**

3. Which tag is used along with <img>, <map> and <area> tag when both methods of Imagemap are used together?
  - a. <a>
  - b. <anchor>
  - c. <href>
  - d. <link>

---

### 3.6. WHAT ARE STYLE SHEETS?

---

Elements that are important for web page are content and presentation. HTML has set of tags that concentrate on the content of the page. It also has set of tags that concentrate on presentation of the content. But still there were many problems faced by web designers. The following techniques were used to improve the presentation of web pages:

- Using proprietary HTML extensions
- Converting text into images
- Using images for white space control

- Use of tables for page layout
- Writing a program instead of using HTML

These techniques considerably increase the complexity of Web pages, offer limited flexibility, suffer from interoperability problems, and create hardships for people with disabilities.

---

### 3.7. WHY ARE SHEETS VALUABLE?

---

Style sheets solve these problems at the same time they replace the limited range of presentation mechanisms in HTML. Style sheets make it easy to specify the amount of white space between text lines, indentation of lines, the colors used for the text and the backgrounds, the font size and style, and a many other details. Style sheets – Documents that provide specifications for how content should look onscreen.

W3C first introduced CSS1 (Cascading Style sheet) which included content's presentation on the web page. Later W3C came up with CSS2 which included: content's presentation, for different media types (aural, print, Braille, projection, etc), capability to specify clipping regions, overflow, visibility, and minimum and maximum widths and heights in the visual formatting model.

---

### 3.8. DIFFERENT APPROACHES TO STYLE SHEETS

---

Style sheets are collections of style information that are applied to plain text. There are different approaches to style sheets. The CSS2 recommendation supports three ways of including style information in a document. These approaches include:

- **Linked Styles** (Global styles) – Style information is read from a separate file (.CSS) that is specified in the <LINK> tag.
- **Embedded Styles** (Page specific style) – Style information is defined in the document head using the <STYLE> and </STYLE> tags.
- **Inline Styles** (Tag specific style) – Style information is placed inside an <HTML> tag and applies to all content between that tag and its companion closing tag using the <STYLE> attribute.

One can use multiple approaches together for implementing style sheets in an HTML document.

---

### 3.9. USING MULTIPLE APPROACHES

---

**Points to remember:**

- Inline styles override both linked style sheets and style information stored in the document head with the <STYLE> tag.

- Styles defined in the document head override linked style sheets.
- Linked style sheets override browser defaults

---

### 3.10. LINKING TO STYLE INFORMATION IN SEPARATE FILE USING <LINK> TAG

---

In this case first we have to set style information in a external style sheet file. So we have to create .CSS file.

#### **Mystyle.css**

```
.p1 {font-face: Arial; color: blue; font-style: italics }
.p2 {letter-spacing: 5pt; color: red; font-style: oblique }
```

We must including Mystyle.css in our web page using <LINK> tag.

At the beginning of the page in the head section will insert following line:

```
<HEAD>
<LINK REL="stylesheet" HREF="Mystyle.css">
</HEAD>
```

Now we can use both class p1 and p2 in all the pages that includes the above <link> tag. Style information once creating in Mystyle.css can be used in almost pages of website. You can set style information that is uniform in all pages for margin, text, alignment, table layout, etc in this approach.

---

### 3.11. SETTING UP STYLE INFORMATION

---

Syntax for specifying a style characteristic has the form:

```
{characteristic: value }
```

Semicolons should separate multiple characteristics / value pair. E.g.,

```
.p1 {font: 12 pt Times New Roman; line-height: 20 pt; color: black }
```

The following table lists the different font and block level style attributes you can assign to a file containing style information.

#### Font and Block Level Characteristics Permitted in Style Sheets

Characteristic	Possible Values
Font-family	Any typeface available to the browser through Windows (the default font is used of one of the specified fonts is not available)

Font-size	Any size in points (pt 0, inches (in), centimeters (cm), or pixels (px)); larger or smaller (relative-size values); xx-small, x-small, small, medium, large, x-large, xx-large (absolute-size values); or a percentage relative to the parent font's size
Font-weight	Normal, bold; bolder, lighter (relative weights)
Font-style	Normal, italic, oblique
Font-variant	Normal, small caps
Color	Any RGB hexadecimal triplet or HTML English-language color name
Background-attachment	Whether the background image stays fixed or scrolls with the content) scroll, fixed
Background-color	Transparent; any RGB hexadecimal triplet or HTML English-language color name
Background-image	None; URL of image file
Background-repeat	Repeat-x (tile background image only in the horizontal direction), repeat-y (tile only in the vertical direction), repeat (tile in both directions), no-repeat (no tiling)
Border-color	Any RGB hexadecimal triplet or HTML English-language color name
Border-style	None, dashed, dotted, solid, double, groove, ridge, inset, outset
Border-bottom-width	Thin, medium, thick: any number of points (pt), inches (in), centimeters (cm), or pixels (px)
Border-left-width	Thin, medium, thick: any number of points (pt), inches (in), centimeters (cm), or pixels (px)
Border-right-width	Thin, medium, thick: any number of points (pt), inches (in), centimeters (cm), or pixels (px)
Border-top-width	Thin, medium, thick: any number of points (pt), inches (in), centimeters (cm), or pixels (px)
Float	Left, right, none; floats positioned content in left or right margin
Padding-bottom	Any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage of the parent element's width
Padding-left	Any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage of the parent element's width

Padding-right	Any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage of the parent element's width
Padding-top	Any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage of the parent element's width
Text-align	Left, center, right, justify
Text-decoration	None, underline, overline, line-through, blink
Text-indent	Any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage relative to the indentation of the parent element
Text-shadow	The shadow offset is required and can be set to any number of points (pt), inches (in), centimeters (cm), or pixels (px); specification of blur radius and shadow color is optional
Text-transform	Capitalize, uppercase, lowercase, none
line-height	Normal; any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage of the font size
Letter-spacing	Normal; any number of points (pt), inches (in), centimeters (cm), or pixels (px)
Word-spacing	Normal; any number of points (pt), inches (in), centimeters (cm), or pixels (px)
Margin-left	Auto; any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage of the parent element's width.
Margin-right	Auto; any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage of the parent element's width
Margin-top	Auto; any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage of the parent element's width
Margin-bottom	Auto; any number of points (pt), inches (in), centimeters (cm), or pixels (px); or a percentage of the parent element's width
Vertical-align	Baseline, sub, super, top, text-top, middle, bottom, text-bottom; or a percentage of the current line-height

### Content Positioning Characteristics Permitted In Style Sheets

<b>Characteristic</b>	<b>Purpose and Possible Values</b>
Position	Specifies how content to be positioned; possible values are static (content cannot be positioned or repositioned), absolute (content is positioned with respect to the upper-left corner of the browser window). And relative (content is positioned with respect to its natural position in the document).
Top	Specifies the vertical displacement of the positioned content; values can be in points (pt), pixels (px), centimeters (cm), or inches (in) and can have negative values (negative value moves content above its reference point on the screen).
Left	Specifies the vertical displacement of the positioned content; values can be in points (pt), pixels (px), centimeters (cm), or inches (in) and can have negative values (negative value moves content above its reference point on the screen).
Clip:rect (x1, y1, x2, y2)	Defines the size of the clipping region (rectangular area in which the positioned content appears); (x1,y1) are the coordinates of the upper-left corner of the rectangle and (x2,y2) are the coordinates of the lower-right corner.
Overflow	Tells the browser how to positioned content that overflows the space allocated for it; possible values are visible, hidden, auto, and scroll.
Visibility	Enables the document author to selectively display or conceal positioned content; possible values are show or hide.
z-index	Permits stacking of positioned content in the browser screen so that content overlaps; z-index is set to an integer value of 0 or higher (content with a smaller z-index will be positioned below content with higher z-index values).

**Using these characteristics we can give various effects to the web page.**

---

### **3.12. EMBEDDED STYLE INFORMATION USING <STYLE> TAG**

---

After creating a “.CSS” file that will be useful to all the web pages of your web site, now concentrate on page specific style. This type of page specific information can be provided using <STYLE> tag in head section of HTML document.



E.g.: After linking Mystyle.css in your website's "client.html", you want the table in this page should have different layout then you can use embedded approach.

```
<HEAD>
<STYLE>
.table1 {border-color: yellow; background-image: client.jpeg;
border-style: dotted}
</STYLE>
</HEAD>
```

Class table1 can be applied to any/multiple table(s) in client.html

```
<BODY>
<TABLE CLASS=table1 ...>
... </TABLE>
...
<TABLE CLASS=table1 ...>
</TABLE>
</BODY>
```

Selectors are patterns used to select the element(s) you want to apply style. Few example of selector are given in the following table:

Selector	Example	Description
<i>.class</i>	.intro	Selects all elements with class="intro"
<i>#id</i>	#f110	Selects the element with id="f110"
<i>Element</i>	P	Selects all <p> elements. One can use any tag/html element i.e. hr, table, etc.
<i>Link</i>	a:link	Selects all unvisited links
<i>:visited</i>	a:visited	Selects all visited links
<i>:active</i>	a:active	Selects the active link
<i>:hover</i>	a:hover	Selects links on mouse over

The following CSS STYLE declaration puts a border around every H3 element in the document and centers it on the page:

```
<HEAD>
<STYLE TYPE="text/css">
H3 {border-width: 2; border: solid; text-align: center}
</STYLE>
</HEAD>
```

To specify that this style information should only apply to H1 elements of a specific class, we modify it as follows:

```
<HEAD>
<STYLE TYPE="text/css">
H1.myclass {border-width: 1; border: solid; text-align: center}
</STYLE>
</HEAD>
<BODY>
<H1 CLASS="myclass"> This H1 is affected by our style </H1>
<H1> This one is not affected by our style </H1>
</BODY>
```

To limit the scope of the style information to a single instance of P, set the id attribute:

```
<HEAD>
<STYLE TYPE="text/css">
#myid {border-width: 1; border: solid; text-align: center}
</STYLE>
</HEAD>
<BODY>
<P ID="myid"> This paragraph is affected by style </H1>
</BODY>
```

If you want all three kinds of links i.e., unvisited, visited, and active to be rendered in the same style:

```
A:link {font-size: 10pt; color: 00FF00; font-decoration: underline}
A:visited {font-size: 10pt; color: 00FF00; font-decoration:
underline}
A:active {font-size: 10pt; color: 00FF00; font-decoration:
underline}
```

**Or**

```
A:link A:visited A:active {font-size: 10 pt; color: 00FF00; font-
decoration: underline}
```

---

### 3.13. INLINE STYLE INFORMATION

---

Till now you have used both lined and embedded approach in your web site. Suppose if you want the in “client.html” the last paragraph should be displayed in center and with 5 pt word spacing.

At the end of client.html, insert following:

```
<P STYLE="align=center; word-spacing: 5pt">
... content of the paragraph ...
</P>
```

This style is specific to only one instance of <P> tag.

**Let us see the progress:**

4. How many approaches are there to style sheet?  
a. 2                      b. 4                      c. 5                      d. 3
5. I want in my website all main headings of 23 font size, sub heading of 18 font size. Which approach of style sheet is best here?  
a. inline                      b. embedded                      c. Linked

**3.14. SUMMARY**

In this chapter we have discussed what Imagemaps are, creation of client side and server Imagemap, using both this method together. We also studied role of style sheet, use of style sheet, different approaches to style sheet.

Answers of check your progress:

1. b                      2. a                      3. a                      4. d                      5. c

**3.15. EXERCISE****3.15.1. Questions**

1. What are Imagemaps?
2. Explain Server side imagemap in detail.
3. Explain in detail how to create map file that is used for server-side Imagemap.
4. Client side imagemap is better than server side image map. Explain.
5. State and explain steps involved in creating client side scripting language.
6. Explain with example how to combine server side and client side imagemap.
7. What are Stylesheets? Explain role of stylesheet in web designing.
8. What are the different approaches to stylesheet?
9. List and explain content positioning characteristics and its value.
10. List and explain font and block level positioning characteristics and its value.
11. Explain Linked styles with example.
12. Explain embedded styles with example.
13. What are selectors? List and explain selectors in detail.
14. Explain inline styles with example.

**3.15.2. Programs**

1. Take up any jpeg/gif image; write code to apply client side imagemap for any three regions.



## TABLES

### Unit Structure

- 4.0. Objective
- 4.1. Introduction to HTML Tables and Their Structure
- 4.2. The TABLE Tags
- 4.3. Alignment Aligning Entire TABLE
- 4.4. Alignment within a Row
- 4.5. Alignment within a Cell
- 4.6. Attributes
- 4.7. Content Summary
- 4.8. Background Color
- 4.9. Adding a Caption
- 4.10. Setting the Width
- 4.11. Adding a Border
- 4.12. Spacing within a Cell
- 4.13. Spacing between the Cells
- 4.14. Spanning Multiple Rows or Columns
- 4.15. Elements That Can Be Placed in a TABLE
- 4.16. TABLE Sections and Column Properties
- 4.17. Tables as a Design Tool
- 4.18. Exercise

---

### 4.0. OBJECTIVE

---

After going through this chapter you will be able to:

- Learn role of HTML tables in developing web pages
- Create different types of html tables
- Use different attributes of <table> tag
- Identify different table tags that makes a table in html

---

## 4.1. INTRODUCTION TO HTML TABLES AND THEIR STRUCTURE

---

Tables are used on websites for two major purposes:

- Arranging information in a table
- Creating a page layout with the use of hidden tables.

Using tables to divide the page into different sections is an extremely powerful tool. Almost all major sites on the web are using invisible tables to layout the pages. The most important layout aspects that can be done with tables are:

- Dividing the page into separate sections (An invisible table is excellent for this purpose)
- Creating menus (Typically with one color for the header and another for the links following in the next lines)
- Adding interactive form fields (Typically a gray area containing a search option)
- Creating fast loading headers for the page (A colored table with a text on it loads like a bullet compared to even a small banner)
- Easy alignment of images that have been cut into smaller pieces
- A simple way to allow text to be written in two or more columns next to each other

The importance of using tables for these layout purposes can't be overrated. However there are a few things to keep in mind when doing so.

Most important is, that the content of a table is not shown until the entire table is loaded. If you have extremely long pages, you should divide it into two or more tables - allowing the user to start reading the upper content while the rest of the page is loading.

---

## 4.2. THE TABLE TAGS

---

Following are the HTML table tags:

Tag	Functionality
<TABLE>	Contains all HTML tags that compose a table
<CAPTION>	Specifies a caption for a table
<THEAD>	Defines the header section of a table
<TFOOT>	Defines the footer section of the table
<TBODY>	Defines the body section of the table
<COLGROUP>	Groups a set of columns so that properties may be assigned to all columns in the group rather than to each one individually

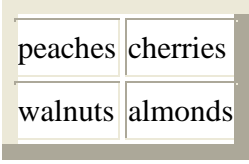
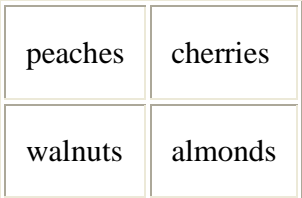
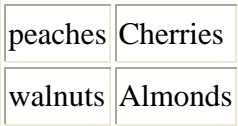
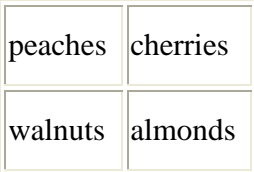
<COL>	Specifies properties for a column or columns within a group
<TR>	Defines a row of a table, table header, table footer, or table body
<TD>	Defines a cell in a table
<TH>	creates a header cell whose contents will be rendered in boldface and with a centered horizontal alignment

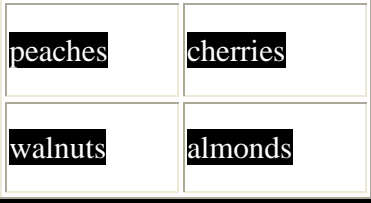

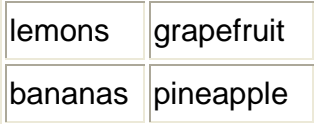
Some table tags which we use frequently are explained with their attributes:

<TABLE> - is the main tag which holds all table related tags in it.

Type: Container

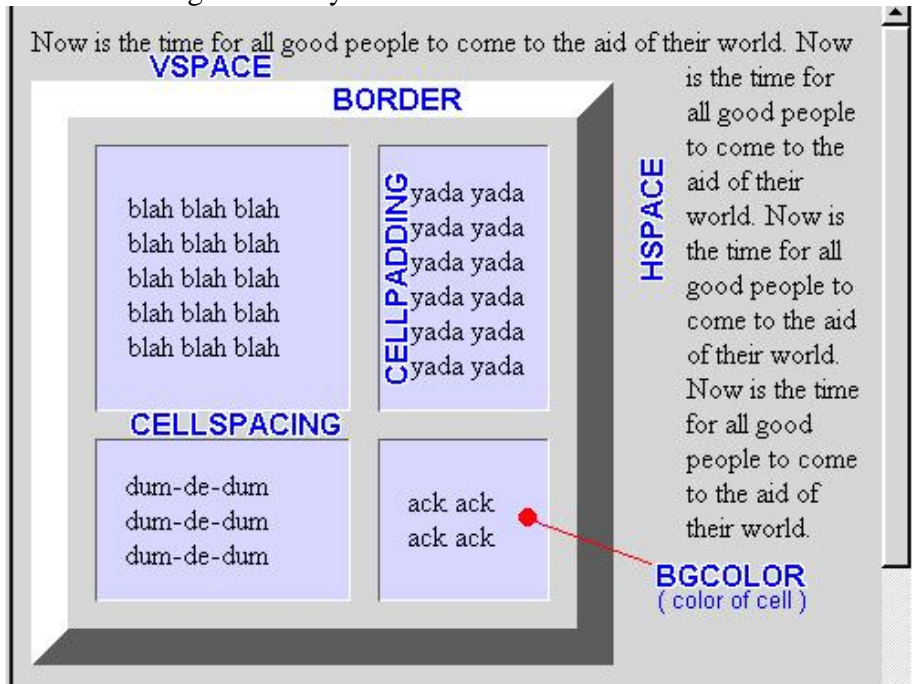
Attributes:

Attribute	Value	Code and output
<u>BORDER</u> : size of border around the table	Integer	<TABLE <b>BORDER=15</b> > 
<u>CELLPADDING</u> : space between the edge of a cell and the contents	Integer	<TABLE BORDER <b>CELLPADDING=10</b> > 
<u>CELLSPACING</u> : space between cells	Integer	<TABLE BORDER <b>CELLSPACING=2</b> > 
<u>WIDTH</u> : width of the table as a whole	% or Pixels	<TABLE BORDER <b>WIDTH=25%</b> > 

<u>BGCOLOR</u> : color of the background	Color Expression	<TABLE BGCOLOR="black"> 										
<u>BACKGROUND</u> : picture to use as background	URL of Image	<TABLE BGROUND="WaterLilies.jpg">										
<u>ALIGN</u> : alignment of table to surrounding text	Left Right Center	<TABLE ALIGN="Left">										
<u>HSPACE</u> : horizontal space between table and surrounding text	Integer	<TABLE HSPACE=10 VSPACE=10>  Yada yada yada. Yada yada yada. Yada yada yada. Yada yada yada. Yada yada yada. Yada yada yada. Yada yada yada. Yada yada yada. Yada yada yada. Yada yada yada. Yada yada yada. Yada yada yada.										
<u>VSPACE</u> : vertical space between table and surrounding text												
<u>HEIGHT</u> : height of the table as a whole	Pixels	<TABLE HEIGHT=200 BORDER=1> 										
<u>FRAME</u> : parts of outside border that are visible	VOID   BOX   BORDER   ABOVE   BELOW   LHS   RHS   HSIDES   VSIDES	<TABLE BORDER=8 FRAME=LHS> <table><thead><tr><th>Name</th><th>Food</th></tr></thead><tbody><tr><td>Starflower</td><td>stir fied tofu</td></tr><tr><td>Miko</td><td>vegetable rice soup</td></tr></tbody></table>	Name	Food	Starflower	stir fied tofu	Miko	vegetable rice soup				
Name	Food											
Starflower	stir fied tofu											
Miko	vegetable rice soup											
<u>RULES</u> : if there should be internal borders	NONE   ALL   COLS   ROWS   GROUPS	<TABLE BORDER=8 RULES=COLS> <table><thead><tr><th>Name</th><th>Food</th></tr></thead><tbody><tr><td>Starflower</td><td>stir fied tofu</td></tr><tr><td>Miko</td><td>vegetable rice soup</td></tr><tr><td>Andy</td><td>hummus</td></tr><tr><td>Ping</td><td>french toast</td></tr></tbody></table>	Name	Food	Starflower	stir fied tofu	Miko	vegetable rice soup	Andy	hummus	Ping	french toast
Name	Food											
Starflower	stir fied tofu											
Miko	vegetable rice soup											
Andy	hummus											
Ping	french toast											

<b>BORDERCOLOR</b> : color of border around the table	Color Expression	<TABLE BORDERCOLOR=BLACK>
<b>SUMMARY</b> : Summary of the purpose of the table	Text	<TABLE SUMMARY="Attributes table">

Picture showing commonly used attributes:



<TR>: designates a table row. Each <TR> element contains one or more <TD> or <TH> elements.

Type: Container

**Attributes:**

Attribute	Value	Code and Output						
<b>ALIGN</b> : horizontal alignment of cell contents	LEFT   CENTER   RIGHT	<TR ALIGN=right> <table border="1"> <tr> <td><b>Fruit</b></td> <td><b>State</b></td> </tr> <tr> <td>watermelon</td> <td>Georgia</td> </tr> <tr> <td>apples</td> <td>Washington</td> </tr> </table>	<b>Fruit</b>	<b>State</b>	watermelon	Georgia	apples	Washington
<b>Fruit</b>	<b>State</b>							
watermelon	Georgia							
apples	Washington							
<b>VALIGN</b> : vertical alignment of cell contents	TOP   MIDDLE   BOTTOM   BASELINE	<TR VALIGN=top> <table border="1"> <tr> <td><b>Fruit</b></td> <td><b>Largest State Producer</b></td> </tr> </table>	<b>Fruit</b>	<b>Largest State Producer</b>				
<b>Fruit</b>	<b>Largest State Producer</b>							



<b>BGCOLOR:</b> background color	Color Expression	<TR BGCOLOR=black>
<b>BACKGROUND:</b> background image	URL of Image	<TR BACKGROUND=rose.gif>
<b>BORDERCOLOR:</b> color of border around each cell	Color Expression	<TR BORDERCOLOR=red>

<TD>: set a single table cell or column  
Type: Container

#### Attributes:

Attribute	Value	Code and Output
<b>ALIGN:</b> horizontal alignment of cell contents	LEFT   CENTER   MIDDLE   RIGHT	<TD ALIGN=center>
<b>VALIGN:</b> vertical alignment of cell contents	TOP   MIDDLE   CENTER   BOTTOM   BASELINE	<TD VALIGN=Middle>
<b>WIDTH:</b> width of cell	% or Pixels	<TD WIDTH=60%>
<b>HEIGHT:</b> height of cell	Pixels	<TD HEIGHT=20>
<b>COLSPAN:</b> number of columns to cover	Integer	<pre>&lt;table&gt; &lt;caption&gt;Student Details&lt;/caption&gt; &lt;tr&gt; &lt;th rowspan="2"&gt;Name&lt;/th&gt; &lt;th colspan="2"&gt;Address&lt;/th&gt; &lt;/tr&gt; &lt;tr&gt; &lt;th&gt;City&lt;/th&gt; &lt;th&gt;Street&lt;/th&gt; &lt;/tr&gt; &lt;tr&gt; &lt;td&gt;Amit&lt;/td&gt; &lt;td&gt;Mumbai&lt;/td&gt; &lt;td&gt;D.N.Road&lt;/td&gt; &lt;/tr&gt; &lt;/table&gt;</pre>
<b>ROWSPAN:</b> number of rows to cover		

		<table border="1"> <tr> <td rowspan="2"><b>Name</b></td> <td colspan="2"><b>Address</b></td> </tr> <tr> <td><b>City</b></td> <td><b>Street</b></td> </tr> <tr> <td>Amit</td> <td>Mumbai</td> <td>D.N.Road</td> </tr> </table>	<b>Name</b>	<b>Address</b>		<b>City</b>	<b>Street</b>	Amit	Mumbai	D.N.Road
<b>Name</b>	<b>Address</b>									
	<b>City</b>	<b>Street</b>								
Amit	Mumbai	D.N.Road								
<u>NOWRAP</u> : don't word wrap										
<u>BGCOLOR</u> : color of the background	Color Expression	<TD BGCOLOR="red">								
<u>BORDERCOLOR</u> : color of border around the table	Color Expression	<TD BORDERCOLOR="red">								
<u>BACKGROUND</u> : picture to use as background	URL of Image	<TD BACKGROUND="water.gif">								

<TH>: It works just like <TD>, except that <TH> indicates that the cell is a header for a column or row. It is identical to <TD> in every way except one: <TH> indicates that the table cell is a header cell, a title for a column or row. <TH> cells are generally rendered with letters in bold.

Type: Container

Attributes: Same as <TD>

E.g., <TABLE BORDER CELLPADDING=4>  
 <TR>  
 <TH>name</TH><TH>extension</TH><TH>department</TH></TR>  
 <TR> <TD>Ajit Sharma</TD><TD>x  
 423</TD><TD>Marketing</TD></TR>  
 <TR> <TD>Ronny Kapoor</TD> <TD>x 454</TD>  
 <TD>Sales</TD></TR>  
 </TABLE>

name	extension	department
Ajit Sharma	x 423	Marketing
Ronny Kapoor	x 454	Sales

**Let us see the progress:**

- Following tag is not a table tag  
 a. tr                      b. td                      c. tc                      d. th
- How many possible values can be assigned to attribute FRAME?  
 a. 9                      b. 4                      c. 7                      d. 6
- Rowspan attribute is used inside which tag?  
 a. tr                      b. td                      c. th                      d. col

---

### 4.3. ALIGNMENT ALIGNING ENTIRE TABLE

---

Align attribute of <table> tag is used to align entire table as discussed earlier it can take values left| right |center.

E.g.,

```
<table align=center>
...
</table>
```

Above code will place the table in the center of the page

---

### 4.4. ALIGNMENT WITHIN A ROW

---

Align and valign attributes of <tr> tag does alignment of the content that is placed within that <tr> i.e. that row. Align can take values: LEFT | CENTER | RIGHT. Valign can take values TOP | MIDDLE | BOTTOM | BASELINE

E.g.,

```
<tr align= RIGHT valign=middle>...</tr>
```

Above code will place the content of the row in middle of the row and on the right of the row.

---

### 4.5. ALIGNMENT WITHIN A CELL

---

Align and valign attributes of <td> or <th> tag does alignment of the content that is placed within that <td> i.e. that cell/column. Align can take values: LEFT | CENTER | RIGHT. Valign can take values TOP | MIDDLE | BOTTOM | BASELINE

E.g.,

```
<td align= RIGHT valign=middle>...</td>
```

Above code will place the content of the cell/column in middle of the column and on the right of the column.

---

### 4.6. ATTRIBUTES

---

Attributes of <Table> tags are discussed in detail [explanation of tags](#)

---

## 4.7. CONTENT SUMMARY

---

Summary attribute of <table> tag allows you to specify the summary of the content of the table.

E.g.,

```
<table summary="Details of F.Y.B.Sc.(IT) "> ... </table>
```

---

## 4.8. BACKGROUND COLOR

---

Bgcolor attribute of <table> tag allows you to give background color to the table. This attribute can take name of the color like pink, blue, red or can take hexadecimal code equivalent of the color.

E.g.,

```
<table bgcolor=black>...</table>
```

Or

```
<table bgcolor=#000000>...</table>
```

---

## 4.9. ADDING A CAPTION

---

<CAPTION>: sets a caption for the table. <CAPTION> goes just after the <TABLE> tag. It does not go inside a <TR>, <TD> or <TH> element. There should be only one <CAPTION> per table.

E.g.,

```
<TABLE BORDER CELLPADDING=4>
<CAPTION>Employee Information</CAPTION>
<TR>
<TH>name</TH><TH>extension</TH><TH>department</TH></TR>
<TR> <TD>Ajit Sharma</TD><TD>x
423</TD><TD>Marketing</TD></TR>
<TR> <TD>Ronny Kapoor</TD> <TD>x 454</TD>
<TD>Sales</TD></TR>
</TABLE>
```

**Output:**

Employee Information

name	extension	department
Ajit Sharma	x 423	Marketing
Ronny Kapoor	x 454	Sales

---

## 4.10. SETTING THE WIDTH

---

Width attribute of <table> tag allows you to set the width of a table. This attribute can value in pixel or in percentage.

E.g.,

```
<table width=60%>...</table>
```

When this table will be displayed in browser it will take 60% of web page. In this case if the screen resolution is 800x600 then table width will be 60% of 800. If the screen resolution is 1024x768 then table width will be 60% of 1024

**Or**

```
<table width=250>...</table>
```

When this table will be displayed in browser it will take fixed width that is 250 pixels irrespective of screen resolution

---

## 4.11. ADDING A BORDER

---

Border attribute of <table> tag allows you to give border to the table.

BORDER establishes the size of the border surrounding the table. The default value is 0, which is an invisible border. If you put in BORDER without a value, it defaults to 1.

Here are some examples of the table above using different border values:

```
<TABLE>
```

```
tv    radio
mobile ac
```

```
<TABLE BORDER=0>
```

```
tv    radio
mobile ac
```

```
<TABLE BORDER=15>
```

tv	radio
mobile	ac

Note that with a BORDER value of 0, the *internal* borders are invisible; with any value 1 and up, they are visible, but do not change size.

---

## 4.12. SPACING WITHIN A CELL

---

CELLPADDING attribute of <table> tag allows you to put space within a cell. CELLPADDING sets the amount of space (both horizontal and vertical) between the cell wall and the contents. The default value for CELLPADDING (i.e., if you don't use the attribute at all) is 1. So, for example, the following examples demonstrate CELLPADDING when it is absent, when it is set to 1, and when it is set to 10.

Code	Output				
<TABLE BORDER>	<table border="1"> <tr> <td>tv</td> <td>radio</td> </tr> <tr> <td>mobile</td> <td>Ac</td> </tr> </table>	tv	radio	mobile	Ac
tv	radio				
mobile	Ac				
<TABLE BORDER CELLPADDING=1>	<table border="1"> <tr> <td>tv</td> <td>radio</td> </tr> <tr> <td>mobile</td> <td>Ac</td> </tr> </table>	tv	radio	mobile	Ac
tv	radio				
mobile	Ac				
<TABLE BORDER CELLPADDING=10>	<table border="1"> <tr> <td>tv</td> <td>Radio</td> </tr> <tr> <td>mobile</td> <td>ac</td> </tr> </table>	tv	Radio	mobile	ac
tv	Radio				
mobile	ac				

---

### 4.13. SPACING BETWEEN THE CELLS

---

CELLSPACING attribute of <table> tag allows you to put space between the cells. CELLSPACING sets the amount of space between the cells of a table. If the borders are visible, CELLSPACING controls the width of the internal borders. So, for example, following examples demonstrate CELLSPACING when it is absent, when it is set to 2, and when it is set to 10.

Code	Output				
<TABLE BORDER>	<table border="1"> <tr> <td>One</td> <td>two</td> </tr> <tr> <td>three</td> <td>four</td> </tr> </table>	One	two	three	four
One	two				
three	four				
<TABLE BORDER CELLSPACING=2>	<table border="1"> <tr> <td>One</td> <td>two</td> </tr> <tr> <td>three</td> <td>four</td> </tr> </table>	One	two	three	four
One	two				
three	four				
<TABLE BORDER CELLSPACING=10>	<table border="1"> <tr> <td>One</td> <td>two</td> </tr> <tr> <td>three</td> <td>four</td> </tr> </table>	One	two	three	four
One	two				
three	four				

---

### 4.14. SPANNING MULTIPLE ROWS OR COLUMNS

---

Attributes of <TD> tag COLSPAN and ROWSPAN are used to span multiple rows and columns.

Table cells can span across more than one column or row. The attributes COLSPAN ("how many across") and ROWSPAN ("how many down") indicate how many columns or rows a cell should take up.

For example, we might want to create header cells for each department in our table of names and phone numbers. In this table, the header cells in the first and fifth rows span across two columns to indicate the department for each group of names.

E.g.,

```
<TABLE BORDER=2 CELLPADDING=4>
<TR> <TH COLSPAN=2>Production</TH> </TR>
<TR> <TD>Reema Shah</TD><TD>1493</TD> </TR>
<TR> <TD>Akshay Barucha</TD><TD>3829</TD> </TR>
<TR> <TD>Beena Das</TD><TD>0283</TD> </TR>
<TR> <TH COLSPAN=2>Sales</TH> </TR>
<TR> <TD>Bhushan Dave</TD><TD>4827</TD> </TR>
<TR> <TD>Amit Patil</TD><TD>7246</TD> </TR>
<TR> <TD>Rohan Pathak</TD><TD>5689</TD> </TR>
</TABLE>
```

**Output:**

Production	
Reema Shah	1493
Akshay Barucha	3829
Beena Das	0283
Sales	
Bhushan Dave	4827
Amit Patil	7246
Rohan Pathak	5689

---

#### 4.15. ELEMENTS THAT CAN BE PLACED IN A TABLE

---

<TABLE>, <CAPTION>, <THEAD>, <TFOOT>, <TBODY>, <COLGROUP>, <COL>, <TR>, <TD>, and <TH> are the elements that can be placed in a HTML TABLE. Some tags we have already discussed others we will discuss in coming sections.

---

#### 4.16. TABLE SECTIONS AND COLUMN PROPERTIES

---

<THEAD>, <TFOOT> and <TBODY> defines header, footer and body section of a table respectively. <THEAD> indicates that a group of rows are the header rows at the top of the table. <TBODY> indicates that a group of rows are body rows. <TFOOT> indicates that a group of rows are the footer rows at the bottom of the table.

The most popular use for these three tags, which are currently only recognized by MSIE 4 and up, is to put borders between groups of rows instead of between every row. For example, suppose you have a table in which you want borders around the top row, the bottom row, and around the entire block of rows in between. You could do that with the following code. Note that in addition to <THEAD>, <TBODY>, and <TFOOT> you also must use <TABLE RULES=GROUPS>:

```
<TABLE CELLPADDING=6 RULES=GROUPS FRAME=BOX>
<THEAD>
<TR><TH>Weekday</TH><TH>Date</TH><TH>Manager</TH>
<TH>Qty</TH></TR>
</THEAD>
<TBODY>
<TR> <TD>Mon</TD> <TD>09/11</TD>
<TD>Kinjal</TD><TD>639</TD></TR>
<TR> <TD>Tue</TD> <TD>09/12</TD>
<TD>Leena</TD><TD>596</TD></TR>
<TR> <TD>Wed</TD> <TD>09/13</TD>
<TD>Ruma</TD><TD>1135</TD></TR>
<TR> <TD>Thu</TD> <TD>09/14</TD>
<TD>Sushant</TD><TD>1002</TD></TR>
<TR> <TD>Fri</TD> <TD>09/15</TD>
<TD>Ram</TD><TD>908</TD></TR>
<TR> <TD>Sat</TD> <TD>09/16</TD>
<TD>Leena</TD><TD>371</TD></TR>
<TR> <TD>Sun</TD> <TD>09/17</TD> <TD>Sushant</TD>
<TD>272</TD></TR>
</TBODY>
<TFOOT>
<TR> <TH ALIGN=LEFT COLSPAN=3>Total</TH>
<TH>4923</TH> </TR>
</TFOOT>
</TABLE>
```

### Output:

Weekday	Date	Manager	Qty
Mon	09/11	Kinjal	639
Tue	09/12	Leena	596
Wed	09/13	Ruma	1135
Thu	09/14	Sushant	1002
Fri	09/15	Ram	908
Sat	09/16	Leena	371
Sun	09/17	Sushant	272
<b>Total</b>			<b>4923</b>



<COLS>: sets properties for a column of table cells. <COL> is an HTML 4.0 tag. Currently only MSIE and Netscape 6 recognize it.

<COL> goes after the <TABLE> tag and before any <TR>, <THEAD>, or <TBODY> elements. (It may go inside a <COLGROUP> element) Each <COL> defines the properties of one column, unless you use SPAN to indicate that it is for more than one column. So the first <COL> sets properties for first column, the second <COL> sets properties for second column, and so on.

Type: Container

**Attributes:**

<p><u>SPAN</u>: how many columns &lt;col&gt; tag affects</p>	<p><i>Integer</i> (default is 1)</p>	<pre>&lt;TABLE BORDER CELLPADDING=5&gt; &lt;COL&gt; &lt;COL STYLE="color:gray" SPAN=2&gt; &lt;TR&gt; &lt;TH&gt;Expense&lt;/TH&gt; &lt;TH&gt;Price&lt;/TH&gt; &lt;TH&gt;Status&lt;/TH&gt; &lt;/TR&gt; &lt;TR&gt; &lt;TD&gt;office suite&lt;/TD&gt; &lt;TD&gt;1,343.11&lt;/TD&gt; &lt;TD&gt;rental&lt;/TD&gt; &lt;/TR&gt; &lt;TR&gt; &lt;TD&gt;cabling&lt;/TD&gt; &lt;TD&gt;1.00&lt;/TD&gt; &lt;TD&gt;installed&lt;/TD&gt; &lt;/TR&gt; &lt;/TABLE&gt;</pre> <table border="1" data-bbox="715 1160 1126 1435"> <thead> <tr> <th>Expense</th> <th>Price</th> <th>Status</th> </tr> </thead> <tbody> <tr> <td>office suite</td> <td>1,343.11</td> <td>rental</td> </tr> <tr> <td>cabling</td> <td>1.00</td> <td>installed</td> </tr> </tbody> </table>	Expense	Price	Status	office suite	1,343.11	rental	cabling	1.00	installed
Expense	Price	Status									
office suite	1,343.11	rental									
cabling	1.00	installed									
<p><u>ALIGN</u>: horizontal alignment</p>	<p>LEFT   CENTER   RIGHT   JUSTIFY</p>	<pre>&lt;TABLE BORDER CELLPADDING=5&gt; &lt;COL ALIGN=LEFT&gt; &lt;COL ALIGN=RIGHT&gt; &lt;COL ALIGN=CENTER&gt; &lt;COL ALIGN=JUSTIFY&gt; &lt;TR&gt; &lt;TH&gt;Name&lt;/TH&gt; &lt;TH&gt;Price&lt;/TH&gt; &lt;TH&gt;Status&lt;/TH&gt; &lt;TH&gt;Comments&lt;/TH&gt; &lt;/TR&gt; more table rows &lt;/TABLE&gt;</pre>									

		<table border="1"> <thead> <tr> <th>Name</th> <th>Price</th> <th>Status</th> <th>Comments</th> </tr> </thead> <tbody> <tr> <td>Compact</td> <td>3500</td> <td>available</td> <td>Nice deal</td> </tr> <tr> <td>Deluxe</td> <td>4000</td> <td>Not available</td> <td>Good view</td> </tr> </tbody> </table>	Name	Price	Status	Comments	Compact	3500	available	Nice deal	Deluxe	4000	Not available	Good view
Name	Price	Status	Comments											
Compact	3500	available	Nice deal											
Deluxe	4000	Not available	Good view											
<b>WIDTH:</b> width of the column	pixels	<pre>&lt;TABLE BORDER CELLPADDING=5&gt; &lt;COL SPAN=3&gt; &lt;COL WIDTH="150px"&gt; &lt;TR&gt; &lt;TH&gt;Name&lt;/TH&gt; &lt;TH&gt;Price&lt;/TH&gt; &lt;TH&gt;Status&lt;/TH&gt; &lt;TH&gt;Comments&lt;/TH&gt; &lt;/TR&gt; more table rows &lt;/TABLE&gt;</pre>												
<b>BGCOLOR:</b> background color of the column	Name of color or equivalent hexadecimal code	<pre>&lt;TABLE BORDER CELLPADDING=5&gt; &lt;COL BGCOLOR="#CCCC99"&gt; more table rows &lt;/TABLE&gt;</pre>												

E.g.,

```
<TABLE BORDER CELLPADDING=5>
<COL>
<COL ALIGN=RIGHT>
<COL STYLE="color:gray">
<TR><TH>Room
Type</TH><TH>Price</TH><TH>Status</TH></TR>
<TR><TD>Delux</TD><TD>4000</TD><TD>available</TD></
TR>
<TR><TD>Compact</TD><TD>2500</TD><TD>not
available</TD></TR>
</TABLE>
```

**Output:**

Room Type	Price	Status
Delux	4000	available
Compact	2500	not available

**Let us see the progress:**

4. Which tag defines one of the table section?  
a. col            b. tbody        c. tr            d. th
5. Which tag is used to group table columns?  
a. col            b. colgroup    c. span        d. cols

---

## **4.17. TABLES AS A DESIGN TOOL**

---

Whenever we are designing web pages which has got multiple topics to be displayed in one page then for better representation of data we can use table tags. Table divides the page into multiple columns & rows. Each column or row can contain different set of information /topic.

As we know whenever data is in tabular format, it is more readable. So while displaying content on the web pages designer can use table for placing text, image, etc. Using tables more contents can be displayed in systematic manner. Use of tables in web pages, makes your page designing simpler still organized.

---

## **4.18. SUMMARY**

---

In this chapter we have seen why tables are used while creating web pages, different table tags, their attributes, effect of different attributes. We also discussed different tags that create table head, body and footer. Here we discussed how to change rendering properties of different columns and to group them together.

Answers to let's check your progress

1. c            2. a            3. b            4. b            5. b

---

## **4.19. EXERCISE**

---

### ***4.19.1. Questions***

1. What is role of tables in web designing?
2. How many table tags are available? Name and explain their use.
3. How to create a table row and how to align content of a row? Explain drawing diagram
4. How to create a table cell/column and how to align content of a cell/column? Explain drawing diagram
5. Explain in detail attributes of <TABLE> tag
6. Explain how columns can be grouped within a table.
7. Explain <COL> and <COLGROUP> tags in detail

### 4.19.2. Programs

1. Write html code to display following output:

Sr. No.	Product Code	Product Description	Price	Quantity	Amount
1	P001	Ceiling Fan	1000	1	1000
2	P007	Electric Burner	2000	1	2000

2. Write html code to display following output:

Yellow	Blue	Red
245	156	453

3. Write html code to display following output:

one	two	three
1	2	3
I	II	III



# FRAMES

## Unit Structure

- 5.0. Objective
- 5.1. Introduction to Frames
- 5.2. Applications
- 5.3. Frames Document
- 5.4. The <FRAMESET> Tag
- 5.5. Nesting <FRAMESET> Tag
- 5.6. Placing Content in Frames with the <FRAME> Tag
- 5.7. Targeting Named Frames
- 5.8. Creating Floating Frames
- 5.9. Using Hidden Frames
- 5.10. Exercise

---

## 5.0. OBJECTIVE

---

After going through this chapter you will be able to:

- Identify role of frames in designing web pages
- Identify different frame tags and their attributes
- Learn how to use frame tags to divide web page in multiple sections
- Explain the role of floating frames
- Explain use of hidden frames

---

## 5.1. INTRODUCTION TO FRAMES

---

Frames allow you to divide the page into several rectangular areas and to display a separate document in each rectangle. Each of those rectangles is called a "frame". Frames are very popular because they are one of the few ways to keep part of the page stationary while other parts change. Frames are also one of the most controversial uses of HTML, because of the way the frames concept was designed, and because many web framed web sites are poorly implemented.

The disadvantages of using frames are:

- Frames are not expected to be supported in future versions of HTML i.e. HTML 5.0
- Frames are difficult to use. (Printing the entire page is difficult).
- The web developer must keep track of more HTML documents

---

## 5.2. APPLICATIONS

---

**Following are the applications of frames:**

- **Page Identification or Context Information:** Information that lets readers know their context, including title, summary, or a special image, works well in frames. Otherwise, scrolling the page hides that information. This is similar to a book where the book title, chapter title, and section title are repeated on headers or footers on every page, not just the first page. Book designers have learned that readers need to know where they are.
- **Main Site Navigation:** Commonly used links, such as links to a home page and major sections, should stay accessible using a frame. Common links can include buttons such as "mailto" link.
- **Document Navigation:** Frames are useful in providing a complete, compact map of the entire document. Maps should be "You Are Here" maps (that indicate the page you are currently viewing) whenever possible. They should be clickable, so that clicking on the representation of a page will follow a link to the page.
- **Local Navigation:** Links to related pages, such as Next/Previous in a list, or sibling button-bars, work very well in frames. Again, "You Are Here" indication is a real plus.
- **Navigating through a Long, Linear Scrolling Page:** A common use for frames is to have one frame that contains a long page and another frame with a list of links to named anchors inside of the other frame. Clicking on a link causes the browser to scroll the long page to the appropriate place. This is done by using <A> tags with HREF attributes pointing to anchors with NAME attributes and with a TARGET attribute pointing to the frame containing the long page.
- **Sidebars:** Sidebars hold material related to the main text. One type of sidebar material that works well in frames are lists of related links.

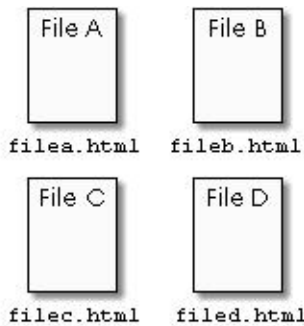
### Frames Document

The frameset file is the file you point your browser to. The frameset file uses `<FRAMESET>` and `<FRAME>` to tell the browser to go get more files to put on the page.

```
<FRAMESET ... >
.
.
.
</FRAMESET>
```

`basicframeset.html`

### The frameset file...



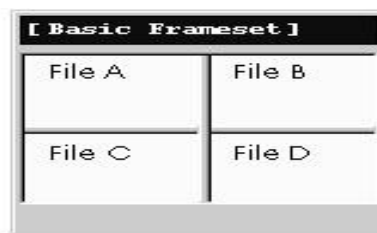
The browser goes out again and retrieves the files which will appear on the page.

**tells the browser to  
go get these four files**

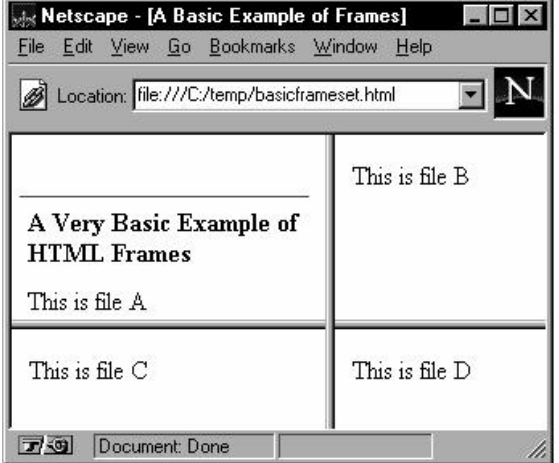
The browser puts all files on one page in separate rectangles ("frames"). The user never sees anything from the original frameset file.

Think of frames as creating a "table of documents" on the page. Like a table, a group of frames has rows and columns. Each cell of the table contains a document which is stored in a separate file. `<FRAMESET>` defines the beginning and end of the table, and how many rows and columns that table will have. `<FRAME>` defines what will go into each cell ("frame") of table.

Let's look in more detail at the example above. The entire contents of `basicframeset.html` (the frameset file) look like this:



**and put them all on  
one page in separate  
rectangles ("frames")**

Code	Output
<pre> &lt;HTML&gt; &lt;HEAD&gt; &lt;TITLE&gt;A Basic Example of Frames&lt;/TITLE&gt; &lt;/HEAD&gt; &lt;FRAMESET ROWS="75%, *" COLS="*, 40%"&gt;   &lt;FRAME SRC="framea.html"&gt;   &lt;FRAME SRC="frameb.html"&gt;   &lt;FRAME SRC="framec.html"&gt;   &lt;FRAME SRC="framed.html"&gt; &lt;/NOFRAMES&gt; &lt;H1&gt;No Frames? No Problem!&lt;/H1&gt;   Take a look at our   &lt;A   HREF="basic.noframes.html"&gt;no- frames&lt;/A&gt; version.   &lt;/NOFRAMES&gt; &lt;/FRAMESET&gt; &lt;/HTML&gt; </pre>	

Here's a line-by-line explanation of each piece of code for the frames:

#### **<FRAMESET**

Start the "table of documents".

#### **ROWS="75%, \*"**

The table should have two rows. The first row should take up 75% of the height of the page; the second should take up the rest.

#### **COLS="\*, 40%">**

The table should also have two columns. The *second* column should take up 40% of the width of the page; the first column should take up the rest.

#### **<FRAME SRC="framea.html">**

#### **<FRAME SRC="frameb.html">**

#### **<FRAME SRC="framec.html">**

#### **<FRAME SRC="framed.html">**

Put the four files into the frames.

#### **<NOFRAMES> ... </NOFRAMES>**

Every framed page should have a no-frames alternative. The `<NOFRAMES>` content should go inside the outermost `<FRAMESET>` tag, usually just before the last `</FRAMESET>`. The most efficient method for no-frames content is to link to a page which is specifically designed for no-frames.

#### **</FRAMESET>**

End the frameset.



There are several other aspects of frames to note from this example:

- `<FRAMESET >` is used instead of the `<BODY >` tag. The frameset file has no content which appears on the page, so it has no need for `<BODY>`, which designates the content of the page. In fact, if you use `<BODY>` (except inside `<NOFRAMES>`), the frames will not appear. Tags in `<HEAD>`, including `<TITLE>`, still has their intended effects.
- Rows and columns are described by a list of widths or heights. For example `COLS="25%, *, 40%"` says that there will be three columns. The first column takes up 25% of the width of the page, the third column takes up 40% of the width of the page, and the asterisk ("`*`") means "whatever is left over".
- You do not explicitly designate start and ending of each row. The browser keeps adding frames until it reaches the number designated by `COLS`, and then starts another row.

**Let's check your progress:**

1. if we have divided our page in two parts using frames how many html pages we have to create ?  
a. 2                      b.3                      c.1                      d.4
2. Can one specify rows and cols attributes of `<frameset>` tag together ?  
a. yes                      b. no

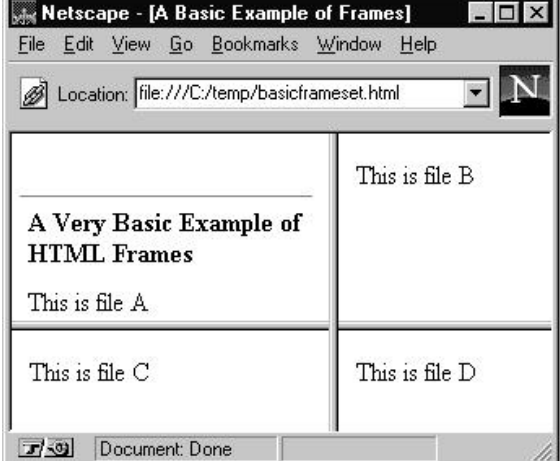
---

### **5.3. THE `<FRAMESET>` TAG**

---

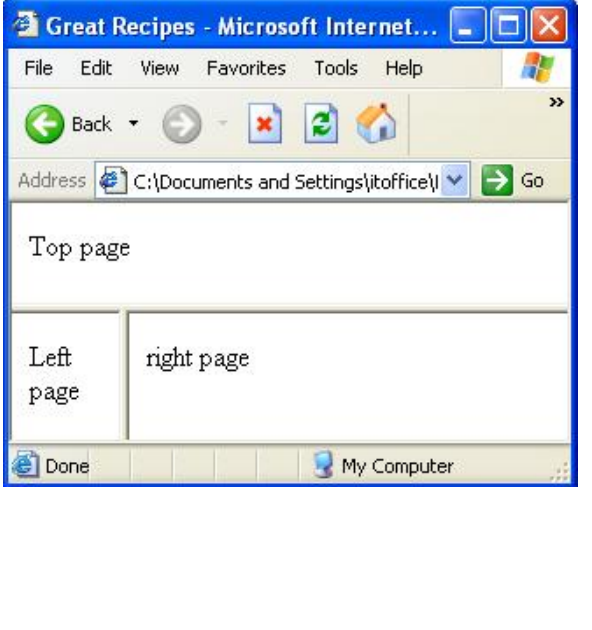
`<FRAMESET>` defines the general layout of a web page that uses frames. `<FRAMESET>` is used in conjunction with `<FRAME>` and `<NOFRAMES>`.

`<FRAMESET>` creates a "table of documents" in which each rectangle (called a "frame") in the table holds a separate document. In its simplest use, `<FRAMESET>` states how many columns and/or rows will be in the "table". You must use either the `COLS` or the `ROWS` attributes or both. For example, this code creates a set of frames that is two columns wide and two rows deep:

Code	Output
<pre> &lt;HTML&gt; &lt;HEAD&gt; &lt;TITLE&gt;A Basic Example of Frames&lt;/TITLE&gt; &lt;/HEAD&gt; &lt;FRAMESET ROWS="75%, *" COLS="*, 40%"&gt;   &lt;FRAME SRC="framea.html"&gt;   &lt;FRAME SRC="frameb.html"&gt;   &lt;FRAME SRC="framec.html"&gt;   &lt;FRAME SRC="framed.html"&gt; &lt;/FRAMESET&gt; &lt;/HTML&gt; </pre>	

<FRAMESET> itself only define how many rows and columns of frames there will be. <FRAME> defines what files will actual go into those frames.

<FRAMESET> can be **nested** within another <FRAMESET> to create a "table within a table". By doing this you can create frames that are not strict grids like in the example above. This set of nested framesets creates the popular "title and sidebar" layout.

Code	Output
<pre> &lt;HTML&gt; &lt;HEAD&gt; &lt;TITLE&gt;Great Recipes&lt;/TITLE&gt; &lt;/HEAD&gt; &lt;FRAMESET ROWS="15%,*"&gt;   &lt;FRAME SRC="top.html" NAME=TITLE SCROLLING=NO&gt;   &lt;FRAMESET COLS="20%,*"&gt;     &lt;FRAME SRC="left.html" NAME=left&gt;     &lt;FRAME SRC="right.html" NAME=right&gt;   &lt;/FRAMESET&gt; &lt;/FRAMESET&gt; &lt;/HTML&gt; </pre>	

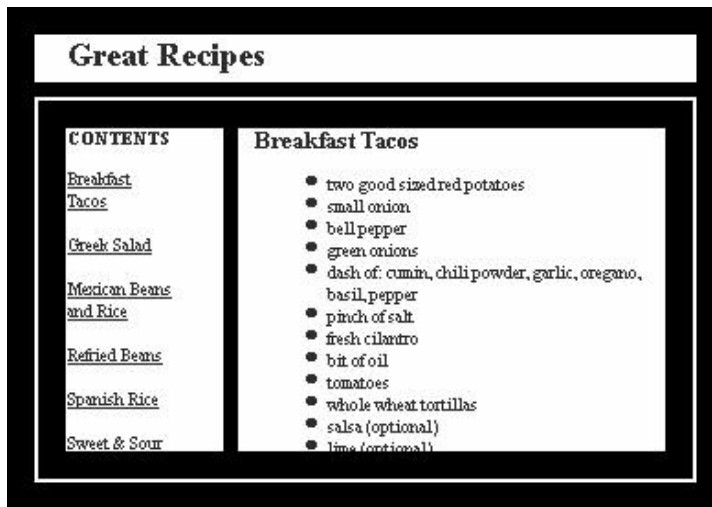
---

## 5.4. NESTING <FRAMESET> TAG

---

Nesting frameset means using one frameset within another frameset. Example shown below illustrates nesting frameset.

The first <FRAMESET > creates a "table" of two rows and only one column (because there is no COLS attribute). The first row in the frameset is filled in by the first<FRAME>. The second row in the frameset is filled in not by a frame but by another <FRAMESET>. This *inner* frameset has two columns, which are filled in by two<FRAMESET>'s.



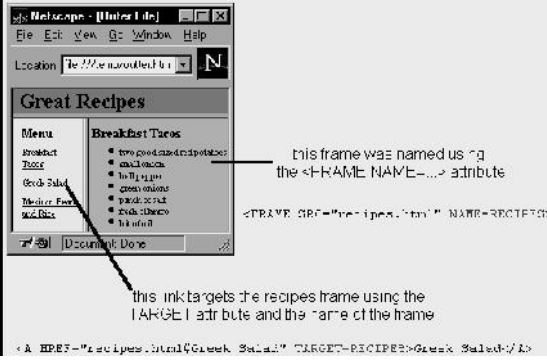
### Code for above output:

```
<HTML>
<HEAD>
<TITLE>Great Recipes</TITLE>
</HEAD>
<FRAMESET ROWS="15%,*">
  <FRAME SRC="recipetitlebar.html" NAME=TITLE
  SCROLLING=NO>
  <FRAMESET COLS="20%,*">
    <FRAME SRC="recipsidebar.html" NAME=SIDEBAR>
    <FRAME SRC="recipes.html" NAME=RECIPES>
  </FRAMESET>
</NOFRAMES>
<H1>Great Recipes</H1>
No frames? No Problem! Take a look at our
<A HREF="recipes.html">no-frames</A> version.
</NOFRAMES>
</FRAMESET>
</HTML>
```

One has to write code for html pages recipetitlebar.html, recipsidebar.html, and recipes.html.

## 5.5. PLACING CONTENT IN FRAMES WITH THE <FRAME> TAG

<FRAME> sets a single frame in the framed page. <FRAME> always goes inside a <FRAMESET> element. The SRC attribute, which is required, indicates the URL of the page that goes in the frame. In most situations you should also use NAME to give the frame a name so that links can target the frame. Attributes used with <FRAME> tag:

Attribute	Description	Example
SRC	SRC = "URL"  SRC indicates the URL to put into the frame.	<FRAME SRC="recipetitlebar.html" NAME=TITLE SCROLLING=NO>
NAME	NAME = "text string"  NAME is used in conjunction with <A TARGET="..."> to indicate which frame the link targets.	
SCROLLING	SCROLLING = YES   NO   AUTO  SCROLLING says if there should be a scroll bar on the right and/or bottom of the frame. YES says there absolutely will be scroll bars, even if they are not needed. NO says there will not be scroll bars, even if they might be needed. AUTO is the default: there will be scroll bars on the side and/or bottom as needed.	<pre>&lt;FRAMESET ROWS="30%,30%,*"&gt;   &lt;FRAME SRC="scrollingYes.html"   SCROLLING=YES&gt;   &lt;FRAME SRC="scrollingNo.html"   SCROLLING=NO&gt;   &lt;FRAME SRC="scrollingAuto.html"   SCROLLING=AUTO&gt; &lt;/NOFRAMES&gt;NOFRAMES stuff &lt;/NOFRAMES&gt; &lt;/FRAMESET&gt;</pre>

<p>NORESIZE</p>	<p>NORESIZE says that the user cannot make the frame bigger or smaller by sliding the borders. Normally the user can put the mouse over the border and move the border left/right or up/down. NORESIZE disables that ability. All borders that run along the frame are affected. For example, this code uses NORESIZE with the frame for the title bar, and so the border along the bottom of the title bar cannot be resized. However, the two frames at the bottom of the page can still be resized by moving the border left and right.</p>	<pre>&lt;HTML&gt; &lt;HEAD&gt; &lt;TITLE&gt;Great Recipes&lt;/TITLE&gt; &lt;/HEAD&gt; &lt;FRAMESET ROWS="20%,*"&gt;   &lt;FRAME SRC="recipetitlebar.html" NAME=TITLE NORESIZE&gt;   &lt;FRAMESET COLS="20%,*"&gt;     &lt;FRAME SRC="recipsidebar.html" NAME=SIDEBAR&gt;     &lt;FRAME SRC="recipes.html" NAME=RECIPES&gt;   &lt;/FRAMESET&gt; &lt;/FRAMESET&gt; &lt;NOFRAMES&gt;NOFRAMES stuff &lt;/NOFRAMES&gt; &lt;/FRAMESET&gt; &lt;/HTML&gt;</pre>
<p>FRAMEBORDER</p>	<p>FRAMEBORDER = YES   1   NO   0</p> <p>By default frames have visible borders between them. Sometimes, however, you want the frames to join directly to each other with no border between them then you can specify frameborder= NO   0</p>	<pre>&lt;FRAMESET ROWS="20%,*" FRAMEBORDER=NO FRAMESPACING=0 BORDER=0&gt;</pre>

BORDERCOLOR	<p>BORDERCOLOR = color expression</p> <p>BORDERCOLOR sets the color of the borders around the frame.</p>	<pre>&lt;FRAMESET ROWS="*,*,40%,*,*"&gt;   &lt;FRAME SRC="bcRow1.html"&gt;   &lt;FRAME SRC="bcRow2.html"&gt;   &lt;FRAME SRC="bcRow3.html" BORDERCOLOR=RED&gt;   &lt;FRAME SRC="bcRow4.html"&gt;   &lt;FRAME SRC="bcRow5.html"&gt; &lt;NOFRAMES&gt;NOFRAMES stuff &lt;/NOFRAMES&gt; &lt;/FRAMESET&gt;</pre>
MARGINWIDTH MARGINHEIGHT	<p>MARGINWIDTH = size in pixels</p> <p>MARGINHEIGHT = size in pixels</p> <p>MARGINWIDTH and MARGINHEIGHT control the inside margins of the document in the frame.</p>	<pre>&lt;FRAMESET ROWS="60%,*,*"&gt;   &lt;FRAME SRC="mwTop.html"&gt;   &lt;FRAME SRC="mwMiddle.html" MARGINWIDTH=1&gt;   &lt;FRAME SRC="mwBottom.html" MARGINWIDTH=50&gt; &lt;NOFRAMES&gt;NOFRAMES stuff &lt;/NOFRAMES&gt; &lt;/FRAMESET&gt;</pre>

Let's check your progress:

3. Can NORESIZE take any value?
  - a. yes
  - b. no
  
4. How many values frameborder attribute can take?
  - a. 4
  - b. 3
  - c. 2
  - d. 1

---

## 5.6. TARGETING NAMED FRAMES

---

Each frame is given a name using `<FRAME NAME="...">`. These names uniquely identify each frame. Using these names, links in other frames can tell the browser that which is the frame the link targets. That is, in which frame content should be displayed. For example, this code creates a framed page, naming the frames TITLE, SIDEBAR, and MAIN:

```
<FRAMESET ROWS="15%,*">
  <FRAME SRC="tfetitle.html" NAME=TITLE
  SCROLLING=NO MARGINHEIGHT=1>
  <FRAMESET COLS="20%,*">
    <FRAME SRC="tfesidebar.html" NAME=SIDEBAR>
    <FRAME SRC="tfemain.html" NAME=MAIN>
  </FRAMESET>
</NOFRAMES>NOFRAMES stuff
</NOFRAMES>
</FRAMESET>
```

To target one of these frames, the link should have a TARGET attribute set to the name of the frame where the linked page should appear. So, for example, this code creates a link to `tfetacos.html` and targets that link to the MAIN frame:

```
<A HREF="tfetacos.html" TARGET=MAIN>my link</A>
```

### Targeting the Whole Window

Eventually in a framed site you want to "break out"... link to a page and have that page take over the entire window. To create this sort of link, we add `TARGET="_top"` to the `<A ...>` tag:

```
<A HREF="wwtarget.html" TARGET="_top">
```

In the previous example we used TARGET to refer to a frame we had named MAIN. In this example, however, we refer to a frame we never named: `"_top"`. We can do this because the outermost frame (that is, the entire window) is already named `"_top"`. `"_top"` is a reserved name which is automatically given to the entire window. So when we say `TARGET="_top"`, we are saying "put the new web page in the entire window". Note that `"_top"` needs to be in all lower-case, it should have quotes around it, and don't forget the underscore ("`_`").

---

## 5.7. CREATING FLOATING FRAMES

---

Floating frames are best described as "frames that you can place like images." These frames can be cut from one place and can be pasted at other place. `<IFRAME>` is the tag that places floating frame on the web page. Attributes that can be used with `<IFRAME>`:

Name	Description
ALIGN	ALIGN ="TOP MIDDLE BOTTOM LEFT RIGHT"  Controls how the floating frame is aligned. TOP, MIDDLE, and BOTTOM alignments make text appear next to the frame, starting at the top, middle, or bottom of the frame. Setting ALIGN to LEFT or RIGHT floats the frame in the left or right margin and allows text to wrap around it
FRAMEBORDER	Setting FRAMEBORDER to 1 turns on the floating frame's borders; setting it to 0 turns them off
HEIGHT	Specifies the height of the floating frame in pixels
MARGINHEIGHT	Specifies size (in pixels) of the top margin of the floating frame
MARGINWIDTH	Specifies size (in pixels) of the left margin of the floating frame
NAME	Gives the floating frame a unique name so it can be targeted by other tags (such as <A>, <FORM>, and <AREA>)
SCROLLING	Controls the presence of scrollbars on the floating frame. Setting SCROLLING to YES makes the browser always put scrollbars on the floating frame, setting it to NO suppresses the scrollbars, and setting it to the default of AUTO lets the browser decide whether the scrollbars are needed or not.
SRC	Tells the browser the URL of the HTML file to load into the floating frame. SRC is a required attribute of the <IFRAME> tag
WIDTH	Specifies the width of the floating frame in pixels

**Example:**

```

<HTML>
<TABLE BORDER=1 WIDTH=400 HEIGHT=400
  BORDERCOLOR=black>
<TR><TD COLSPAN=2 ALIGN=center>Example of Floating
  Frame</TD></TR>
<TR>
<TD WIDTH=20%>&nbsp;   </td>
<TD WIDTH=80%>

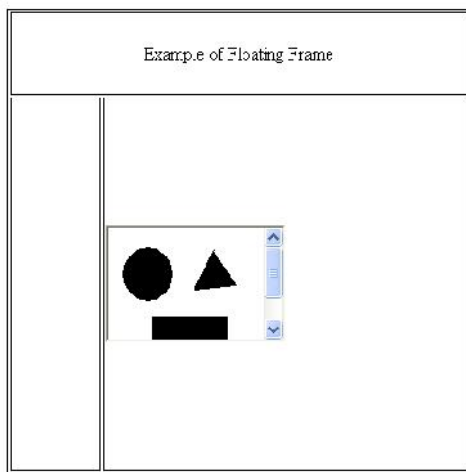
```



```

<IFRAME SRC="shapes.gif" WIDTH="50%" HEIGHT="100"
ALIGN=LEFT
SCROLLING="auto" NAME="floater" FRAMEBORDER=1>
Your browser does not support floating frames.
</IFRAME>
</TD>
</TR>
</TABLE>
</HTML>

```

**Output:****Code Explanation:**

In the above example we have drawn a table with two rows and two columns. In first row colspan is done of two columns. In second row first column has no content but the second column has a floating frame which is placed using <IFRAME> tag. In <IFRAME> tag src attribute tells the browser what is supposed to be displayed in floating frame i.e. an image or an html page. Width and height attribute decides the size of floating frame. Scrolling attribute is set to auto that specifies if content size is bigger than the floating frame size than display scroll bar otherwise do not display scrollbars. Name attribute gives unique name to your floating frame. If your browser do not support <IFRAME> then content within starting and ending of the <IFRAME> tag is displayed. In our case the sentence “Your browser does not support floating frames.” will be displayed.

Let’s check your progress:

5. Can floating frame be placed in a frames page?
  - a. yes
  - b. no

---

## 5.8. USING HIDDEN FRAMES

---

In web pages whenever we are using frames, we assign how much % of the total window it is going to occupy. This is done by specifying rows and cols attribute (as discussed earlier). When Hidden frames are the one which will be occupying 0% of window so they will be seen by user (on web page), but the content of the frame will be part of the source code and will be padded on to the next page.

```
<frameset rows="30%,70%">
<frame src="v1.html" name=visual1>
<frame src="v2.html" name=visual2>
<frame src="hidden.html" name=hidden>
</frameset>
```

In the above code we can see that third frame is not given any % of window. So it will not appear on the web page. So, the third frame is considered as hidden frame.

---

## 5.9. SUMMARY

---

In this chapter we have discussed advantages and disadvantages of using frames. <frame> and <frameset> tags are used to put framed content on a page. We also discussed attributes that these tags take, how nested frames can be created? We learnt to place a floating frame on web page.

---

## 5.10. ANSWER TO LET'S CHECK YOUR PROGRESS

---

1. b                      2. b                      3.b                      4.a                      5.a

---

## 5.11. EXERCISE

---

### 5.11.1. Questions

1. Explain role of frames in web page. Or Explain advantages of using frames in web pages.
2. Explain <frameset> tag with all its attributes.
3. Explain <frame> tag with all its attributes.
4. Explain the tag that will place floating frame on the web.
5. What happens when your browser does not support frames? How a web designer solves this problem?

### 5.11.2. Programs

1. Write html code for a page that is divided into two rows using frames. First row/frame displays logo and name of the company. Second row/frame is divided into two parts (using frames). First part has different links and second part is kept for display.
2. Write html code for the following output:

<b><u>Alpha</u></b> <b><u>Electrical</u></b>	<b>About Us</b>  Details of the company: When it was founded: 1998 Products it deals with: Electrical Branches: in four metro cities
<b>Latest News:</b> <ul style="list-style-type: none"> <li>• Launch of new product next week</li> <li>• Bonus to employees</li> </ul>	

3. Write code for the following output:

<b>XYZ Corporation Limited</b>		
<u>About Us</u> <u>Clients</u> <u>Branches</u> <u>Contact Us</u>		List of products:
<b>Disclaimer   @copyright</b>		



# FORMS

## Unit Structure

- 6.0. Objective
- 6.1. Creating Forms
- 6.2. The <FORM> Tag
- 6.3. Named Input Fields
- 6.4. The <INPUT> Tag
- 6.5. Multiple Lines Text Windows
- 6.6. Drop Down and List Boxes
- 6.7. Hidden
- 6.8. Text
- 6.9. Text Area
- 6.10. Password
- 6.11. File Upload
- 6.12. Button
- 6.13. Radio
- 6.14. Checkbox
- 6.15. Select
- 6.16. Option
- 6.17. Forms and Scripting
- 6.18. Action Buttons
- 6.19. Labeling Input Files
- 6.20. Grouping Related Fields
- 6.21. Disabled and Read Only Fields
- 6.22. Form Field Event Handlers
- 6.23. Passing Form Data
- 6.24. Summary
- 6.25. Exercise

---

## 6.0 OBJECTIVE

---

After going through this chapter you will be able to:

- Learn how to place form in a web page
- Explain how to put different form elements like text field, password, file upload field, submit and reset button, checkbox, radio button, select field
- Learn how to validate a form using script
- Learn how to use event handler with forms
- Learn how to pass form data to another page

---

## 6.1 CREATING FORMS

---

Forms are the most popular way to make web pages interactive. Like forms on paper, a form on a web page allows the user to enter requested information and submit it for processing. (Fortunately, forms on a web page are processed much faster.)

For creating form in a web page we use `<FORM>` tag. `<FORM>` tag is a container tag which can have many other form elements in it like text fields, radio buttons, checkboxes, etc.

---

## 6.2 THE `<FORM>` TAG

---

`<FORM >` indicates the beginning of a form. All other form tags go inside `<FORM>`. In its simplest use, `<FORM>` can be used without any attributes.

Attributes that can be used with form tags are as follows:

Name	Description	Example
URL	ACTION gives the URL of the program which will process this form. For example, the CGI program "MyCGI" is located at <code>../cgi-bin/mycgi.pl</code> (you can go directly to that URL). This form uses "MyCGI":	<pre>&lt;FORM ACTION="../cgi-bin/mycgi.pl"&gt; favorite color: &lt;INPUT NAME=COLOR&gt;&lt;BR&gt; &lt;INPUT TYPE=SUBMIT&gt; &lt;/FORM&gt;</pre> <p>When you click submit, your browser sends the form data to the CGI indicated in ACTION.</p>

METHOD	<p>METHOD = GET   POST</p> <p>METHOD specifies the method of transferring the form data to the web server. METHOD can be either GET or POST. Each method has its advantages and disadvantages.</p> <p>GET sends the data as part of the URL.</p> <p>POST is the preferred method for sending lengthy form data. When a form is submitted POST the user does not see the form data that was sent.</p>	<pre>&lt;FORM METHOD=GET ACTION="../cgi- bin/mycgi.pl"&gt; town: &lt;INPUT NAME="town"&gt;&lt;BR&gt; &lt;INPUT TYPE=SUBMIT&gt; &lt;/FORM&gt;</pre> <p>The value entered in the "town" field is tacked on to the end of the CGI's URL like this:</p> <pre>../cgi- bin/mycgi.pl?town=West+R ochester</pre> <pre>&lt;FORM METHOD=POST ACTION="../cgi- bin/mycgi.pl"&gt;</pre>
NAME	<p>NAME = "text string"</p> <p>NAME gives a name to the form. This is most useful in scripting, where you frequently need to refer to the form in order to refer to the element <i>within</i> the form.</p>	<pre>&lt;SCRIPT&gt; &lt;!-- function Circle_calc_ii() { var CircleRadius = document.MyCircleForm.Ci rcle_radius.value; if (CircleRadius &gt;= 0) { document.MyCircleForm.Ci rcle_circumference.value = 2 * Math.PI * CircleRadius ;  document.MyCircleForm.Ci rcle_area.value = Math.PI * Math.pow(CircleRadius, 2) ; } else { document.MyCircleForm.Ci rcle_circumference.value = ""; }</pre>

		<pre> document.MyCircleForm.Circle_area.value = "";     } } // --&gt; &lt;/SCRIPT&gt;  &lt;FORM NAME="MyCircleForm"&gt; &lt;TABLE BORDER CELLPADDING=3&gt; &lt;TR&gt; &lt;TD&gt;&lt;NOBR&gt;radius: &lt;INPUT NAME="Circle_radius" SIZE=4&gt;&lt;/NOBR&gt;&lt;/TD&gt; &lt;TD&gt;&lt;INPUT TYPE=BUTTON OnClick="Circle_calc_ii(this.form);" VALUE="calculate"&gt;&lt;/TD &gt; &lt;TD ALIGN=RIGHT BGCOLOR="#AACCFF"&gt;     &lt;NOBR&gt;circumference: &lt;INPUT NAME="Circle_circumference" SIZE=9&gt;&lt;/NOBR&gt;&lt;BR&gt;     &lt;NOBR&gt;area: &lt;INPUT NAME="Circle_area" SIZE=9&gt;&lt;/NOBR&gt;&lt;/TD&gt; &lt;/TR&gt;  &lt;/TABLE&gt; &lt;/FORM&gt; </pre>
ENCTYPE	<p>ENCTYPE = "multipart/form-data"   "application/x-www-form-urlencoded"   "text/plain"</p> <p>ENCTYPE determines how the form data is encoded. Whenever data is transmitted from one place</p>	

	<p>to another, there needs to be an agreed upon means of representing that data. Music is translated into written music notation, English is written using letters and punctuation. Similarly, there needs to be an agreed on way of presenting the form data so it's clear that, for example, there is a field called "email" and its value is "abc@docs.com".</p> <p>In most cases you will not need to use this attribute at all. The default value is "application/x-www-form-urlencoded", which is sufficient for almost any kind of form data. The one exception is if you want to do file uploads. In that case you should use "multipart/form-data".</p>	
TARGET	<p>TARGET = "_blank"   "_parent"   "_self"   "_top"   frame name</p> <p>TARGET indicates which frame in a set of frames to send the results to, and works just like &lt;A TARGET="..."&gt;. This attribute can be used so that the form is always visible even as the form results are displayed and redisplayed.</p>	<pre>&lt;FORM   TARGET="TargetFrame"   ACTION="../cgi- bin/mycgi.pl"&gt;</pre>
onSubmit	<p>onSubmit = "script command(s)"</p> <p>onSubmit is a scripting event that occurs when the user attempts to submit the form. onSubmit can be used to do some error checking on the form data, and to cancel the submit if an error is found.</p> <p>Note that in order to cancel the submit event, the onSubmit should be in the form onSubmit="return expression". "return" indicates that value of</p>	<p>This &lt;FORM&gt; tag calls a Javascript function to check the form data:</p> <pre>&lt;FORM   ACTION="../cgi- bin/mycgi.pl"   NAME="testform"   onSubmit="return TestDataCheck()"&gt;</pre>



	the expression should be returned to submit routine. If expression evaluates to false, the submit routine is cancelled; if it is true, the submit routine goes forward.	
onReset	onReset = "script command(s)"  onReset runs a script when the user resets the form. If onReset returns false, the reset is cancelled.	

**Let's check your progress:**

1. Name given to form tag is used by \_\_\_\_\_  
 a. form elements      b. script      c. form      d. table

---

### 6.3 NAMED INPUT FIELDS

---

Whenever we are placing form on a web page for example registrations form. We require various form elements in that form. Each form element should be given a unique name. This unique name will be helpful to us when we are referring these elements in a scripting code.

```
<SCRIPT>
<!--
function Circle_calc_ii()
{
  var CircleRadius = document.MyCircleForm.Circle_radius.value;
  if (CircleRadius >= 0)
  {
    document.MyCircleForm.Circle_circumference.value = 2 * Math.PI
* CircleRadius ;
    document.MyCircleForm.Circle_area.value = Math.PI *
Math.pow(CircleRadius, 2) ;
  }
  else
  {
    document.MyCircleForm.Circle_circumference.value = "";
    document.MyCircleForm.Circle_area.value = "";
  }
}
// -->
```

```

</SCRIPT>
<FORM NAME="MyCircleForm">
<TABLE BORDER CELLPADDING=3>
<TR>
<TD>radius: <INPUT type= text NAME="Circle_radius" SIZE=4></TD>
<TD><INPUT TYPE=BUTTON onClick="Circle_calc_ii(this.form);"
VALUE="calculate"></TD>
<TD ALIGN=RIGHT BGCOLOR="#AACCFF">
    circumference: <INPUT type= text NAME="Circle_circumference"
SIZE=9><BR>
    area: <INPUT type= text NAME="Circle_area" SIZE=9></TD>
</TR>
</TABLE>
</FORM>

```

**Code Explanation:**

In above code we have three text fields each one has got distinct name i.e. Circle\_radius, Circle\_circumference, Circle\_area. When we want to refer to these fields in our scripting code, we write as follows:

```

document.MyCircleForm.Circle_radius.value
document.MyCircleForm.Circle_circumference.value = 2 * Math.PI *
CircleRadius ;
document.MyCircleForm.Circle_area.value = Math.PI *
Math.pow(CircleRadius, 2);

```

---

## 6.4 THE <INPUT> TAG

---

<INPUT> creates the data entry fields on an HTML form. Attributes that can be used with <INPUT> tag are as follows:

Name	Description	Example
TYPE	TYPE = TEXT   CHECKBOX   RADIO   PASSWORD   HIDDEN   SUBMIT   RESET   BUTTON   FILE   IMAGE TYPE establishes what type of data entry field this is.	
NAME	NAME assigns a name to the input field, and is required in most circumstances. In forms which use scripts, the name of the input field is sent to the script.	<FORM ACTION="../cgi-bin/mycgi.pl" onSubmit="return (this.email.value != ")" > email: <INPUT

	<p>The input object is in the elements collection of the form object, and can be referred to by its name using dot notation.</p>	<pre>NAME="email"&gt; &lt;P&gt;&lt;INPUT TYPE=SUBMIT VALUE="submit"&gt; &lt;/FORM&gt;</pre> <p>In this example, we use the this.form.email to refer to the email input field.</p>
<p>VALUE</p>	<p>VALUE sets the value for the input field. VALUE sets the default values for text and password fields, sets the button text in submit, reset and plain buttons, sets the values of the choices in radio buttons, sets the permanent values of hidden fields, and has no effect on file, and image fields.</p>	<pre>&lt;FORM ACTION="../cgi- bin/mycgi.pl"&gt; name: &lt;INPUT TYPE=TEXT NAME="realname" VALUE="wisnesky"&gt;&lt; BR&gt; password: &lt;INPUT TYPE=PASSWORD NAME="realname" VALUE="pacman"&gt; &lt;P&gt;&lt;INPUT TYPE=SUBMIT VALUE="submit"&gt; &lt;/FORM&gt;</pre> <p><b>Output:</b></p> <p>name: <input type="text" value="wisnesky"/></p> <p>password: <input type="password" value="*****"/></p> <p><input type="submit" value="submit"/></p>
<p>SIZE</p>	<p>SIZE = integer</p> <p>SIZE sets how wide a text or password field should be. It has no effect on any other type of field.</p> <p>SIZE does not set the maximum length of what can be typed in</p>	<pre>&lt;FORM ACTION="../cgi- bin/mycgi.pl"&gt; age:&lt;INPUT TYPE=TEXT NAME="age" SIZE=2 &gt;&lt;BR&gt; first name: &lt;INPUT TYPE=TEXT NAME="first" SIZE=10&gt;&lt;BR&gt; last name: &lt;INPUT TYPE=TEXT</pre>

		<pre>NAME="last" SIZE=30&gt;&lt;BR&gt; cosmic plane of origin:&lt;BR&gt; &lt;INPUT TYPE=TEXT NAME="plane" SIZE=70&gt; &lt;P&gt;&lt;INPUT TYPE=SUBMIT VALUE="submit"&gt; &lt;/FORM&gt;</pre> <p><b>Output:</b></p> <p>age: <input type="text"/></p> <p>first name: <input type="text"/></p> <p>last name: <input type="text"/></p> <p>cosmic plane of origin: <input type="text"/></p> <p><input type="submit"/></p>
MAXLENGTH	<p>MAXLENGTH = integer</p> <p>MAXLENGTH sets the maximum number of characters for text or password fields</p>	<pre>account ID: &lt;INPUT TYPE=TEXT NAME="accountID" MAXLENGTH=4&gt;&lt;BR &gt; password: &lt;INPUT TYPE=PASSWORD NAME="password" MAXLENGTH=8&gt;</pre> <p>In account ID field we can enter maximum 4 characters and in password field maximum 8 characters.</p>
CHECKED	CHECKED indicates that a radio button or checkbox should be on when the form first loads.	<pre>&lt;FORM ACTION="../cgi- bin/mycgi.pl"&gt; &lt;INPUT TYPE=CHECKBOX NAME="maillist" CHECKED&gt;Yes! Put</pre>

		<pre> me on the list!&lt;BR&gt; What color would you like?&lt;BR&gt; &lt;INPUT TYPE=RADIO NAME="color" VALUE="green" &gt;Green&lt;BR&gt; &lt;INPUT TYPE=RADIO NAME="color" VALUE="red" &gt;Red&lt;BR&gt; &lt;INPUT TYPE=RADIO NAME="color" VALUE="blue" CHECKED &gt;Blue&lt;BR&gt; &lt;INPUT TYPE=RADIO NAME="color" VALUE="brown" &gt;Brown &lt;P&gt;&lt;INPUT TYPE=SUBMIT VALUE="submit"&gt; &lt;/FORM&gt; <b>Output:</b> <input checked="" type="checkbox"/> Yes! Put me on the list! What color would you like? <input type="radio"/> Green <input type="radio"/> Red <input checked="" type="radio"/> Blue <input type="radio"/> Brown <input type="submit" value="submit"/> </pre>
BORDER	<p>BORDER = integer</p> <p>BORDER is used for image submit buttons. BORDER indicates if there should be a visible border around the image. BORDER only has an effect in Netscape. MSIE does not put any visible border around image submits.</p>	

	<p>When you use the image type of input, you can use many of the same attributes as with &lt;IMG &gt;, including:</p> <p>SRC = "image URL"  HEIGHT, WIDTH, ALT,  HSPACE = integer,  VSPACE = integer,  ALIGN = LEFT   RIGHT    TOP   TEXTTOP   MIDDLE    ABSMIDDLE   CENTER    BOTTOM   ABSBOTTOM    BASELINE</p>	
DISABLED, READONLY	<p>READONLY and DISABLED both remove the functionality of the input field, but to different degrees.</p> <p>READONLY locks the field: the user cannot change the value.</p> <p>DISABLED does same thing but takes it further: user cannot use the field in any way, not to highlight the text for copying, not to select the checkbox, not to submit the form. In fact, a disabled field is not even sent if the form is submitted.</p>	<pre>&lt;INPUT NAME="realname" VALUE="Hi There" READONLY&gt;</pre> <pre>&lt;INPUT NAME="realname" VALUE="Hi There" DISABLED&gt;</pre>
ACCESSKEY	<p>ACCESSKEY = "text string"</p> <p>ACCESSKEY specifies a shortcut key to go directly to the input field. The key is pressed along with the ALT key. For button style fields, using the key is like pressing the button.</p>	<pre>&lt;INPUT TYPE=SUBMIT ACCESSKEY="g" VALUE="Go!"&gt;</pre>
TABINDEX	<p>TABINDEX = integer</p> <p>Normally, when the user tabs from field to field in a form (in a browser that allows tabbing, not all browsers do) the order is the order the fields appear in the HTML code.</p> <p>However, sometimes you want the tab order to flow a little differently. In that case, you</p>	<pre>&lt;FORM ACTION="../cgi- bin/mycgi.pl" METHOD=POST&gt; &lt;TABLE BORDER CELLPADDING=3 CELLSPACING=5 BGCOLOR="#FFFFCC" "&gt;</pre>

	<p>can number the fields using TABINDEX. The tabs then flow in order from lowest TABINDEX to highest.</p>	<pre> &lt;TR&gt;   &lt;TD&gt;name: &lt;INPUT NAME="realname" TABINDEX=1&gt;&lt;/TD&gt;   &lt;TD ROWSPAN=3&gt;commen ts&lt;BR&gt;   &lt;TEXTAREA COLS=25 ROWS=5 TABINDEX=4&gt; &lt;/TEXTAREA&gt;&lt;/TD&gt;&lt;&lt; /TR&gt; &lt;TR&gt; &lt;TD&gt;email: &lt;INPUT NAME="email" TABINDEX=2&gt;&lt;/TD&gt;&lt;&lt; /TR&gt; &lt;TR&gt; &lt;TD&gt;department: &lt;SELECT NAME="dep" TABINDEX=3&gt;   &lt;OPTION VALUE=""&gt;...   &lt;OPTION VALUE="mkt"&gt;Marketi ng   &lt;OPTION VALUE="fin"&gt;Finance   &lt;OPTION VALUE="dev"&gt;Develo pment   &lt;OPTION VALUE="prd"&gt; Production&lt;/SELECT&gt;&lt;&lt; /TD&gt;&lt;/TR&gt; &lt;/TABLE&gt; &lt;/FORM&gt; </pre>
--	---	--

**Let's check your progress:**

2. How form elements can be placed using <INPUT> tag?
  - a. 8
  - b. 7
  - c. 9
  - d. 10
  
3. A \_\_\_\_\_ field is not even sent if the form is submitted
  - a. disabled
  - b. enabled
  - c. readonly
  - d. blank

---

## 6.5 HIDDEN

---

HIDDEN indicates that the field is invisible and the user never interacts with it. The field is still sent to the script, and scripts can also use the hidden field. HIDDEN is commonly used as output of a script which creates a new form for more input. For example, a web site which facilitates online discussions may use a hidden field to keep track of which message is being responded to:

```
<H2>Your Reply</H2>
<FORM METHOD=POST ACTION=" ../cgi-bin/mycgi.pl">
<INPUT TYPE=HIDDEN NAME="postingID" value="98765">
name: <INPUT NAME="realname" SIZE=30><BR>email: <INPUT
NAME="email"><BR>
subject: <INPUT NAME="subject" VALUE="Re: Hamlet and hesitation"
SIZE=30>
<P>comments:<BR>
<TEXTAREA NAME="comments" COLS=50 ROWS=10
WRAP=VIRTUAL>
Joe Smiley wrote:
: I think Hamlet doesn't act because if he does, the play's over.
</TEXTAREA>
<P><INPUT TYPE=SUBMIT VALUE="Send It!">
</FORM>
```

### Output:

The screenshot shows the rendered HTML code above. It features a form with three input fields: 'name:', 'email:', and 'subject: Re: Hamlet and hesitation'. Below these is a 'comments:' section with a text area containing the text: 'Joe Smiley wrote: : I think Hamlet doesn't act because if he does, the play's over.' At the bottom is a 'Send It!' button.

### Code Explanation:

In the above code PostID field is hidden so in the output you will not be able to see that field. But definitely the value of PostID is passed to the next page. So values which you want to pass to next page but without displaying to user then you can hidden field.



Let's check your progress:

4. Which fields are not visible to user on web page?  
 a. readonly    b. hidden    c. disabled    d. visual

---

## 6.6 TEXT

---

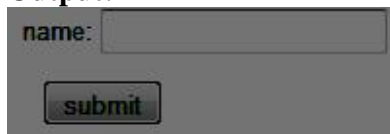
TEXT creates a text entry field (the most popular type of data entry field):

```
<FORM ACTION="../cgi-bin/mycgi.pl">
name: <INPUT TYPE=TEXT NAME="realname">
<P><INPUT TYPE=SUBMIT VALUE="submit">
</FORM>
```

OR

```
<FORM ACTION="../cgi-bin/mycgi.pl">
<!-- Note absence of TYPE attribute -->
name: <INPUT NAME="realname">
<P><INPUT TYPE=SUBMIT VALUE="submit">
</FORM>
```

**Output:**



The screenshot shows a web browser window with a form. On the left, the text 'name:' is displayed. To its right is a rectangular text input field. Below the input field is a button with the word 'submit' written on it.

The behavior of TEXT fields can be modified using these attributes:

VALUE: set an initial value for the field

SIZE: how wide the field should be

MAXLENGTH: the maximum number of characters the user can enter

---



## 6.7 TEXT AREA

---

<TEXTAREA> indicates a form field where the user can enter large amounts of text (i.e. multiline input). In most respects, <TEXTAREA> works like an <INPUT> field. It can have a name, a default value, script events such as onChange, and is sent to a scripting code as a name/value pair. One main difference is that <TEXTAREA> is a container tag.

<TEXTAREA> takes following attributes:

| Name | Description   | Example |
|------|---|---------|
| NAME | NAME = "text string"<br>NAME sets the name of the field for use in scripting. This attribute works just like <INPUT NAME="..."> |         |

|                              |  |  |
|------------------------------|--|--|
| <p>COLS &amp; ROWS</p>       | <p>COLS indicate how many characters (not pixels) wide the text area should be. ROWS indicate how many rows should be in the text area. Both attributes are required in the &lt;TEXTAREA&gt; tag. These attributes do not set any limit on how much can be typed in, just how much of the textarea is visible</p>  | <pre>&lt;TEXTAREA NAME="few" COLS=10 ROWS=2&gt;&lt;/TEXTAREA&gt;</pre> <p><b>Output:</b></p>  <pre>&lt;TEXTAREA NAME="some" COLS=50 ROWS=5&gt;&lt;/TEXTAREA&gt;</pre> <p><b>Output:</b></p>    |
| <p>DISABLED<br/>READONLY</p> | <p>DISABLED and READONLY work exactly like the corresponding attributes for &lt;INPUT&gt;. Please see &lt;INPUT DISABLED&gt; and &lt;INPUT READONLY&gt;.</p>   |  |
| <p>TABINDEX</p>              | <p>TABINDEX = integer</p> <p>TABINDEX is supported by MSIE 4.x and higher and Netscape 6.</p> <p>Normally, when the user tabs from field to field in a form (in a browser that allows tabbing, not all browsers do) the tab order is the order the fields appear in the HTML code.</p> <p>However, sometimes you want the tab order to flow a little differently. In that case, you can number the fields using TABINDEX. The tabs then flow in order from lowest TABINDEX to highest.</p> | <pre>&lt;TABLE BORDER CELLPADDING=3 CELLSPACING=5 BGCOLOR="#ffffcc"&gt; &lt;TR&gt;&lt;TD&gt;name: &lt;INPUT NAME="realname" TABINDEX=1&gt;&lt;/TD&gt; &lt;TD ROWSPAN=3&gt;comments&lt;BR &gt; &lt;TEXTAREA COLS=25 ROWS=5 TABINDEX=4&gt;&lt;/TEXTARE A&gt;&lt;/TD&gt;&lt;/TR&gt; &lt;TR&gt; &lt;TD&gt;email: &lt;INPUT NAME="email" TABINDEX=2&gt;&lt;/TD&gt;&lt;/TR&gt; &lt;TR&gt; &lt;TD&gt;department: &lt;SELECT NAME="dep" TABINDEX=3&gt;</pre> |

		<pre> &lt;OPTION VALUE=""&gt;... &lt;OPTION VALUE="mkt"&gt;Marketing &lt;OPTION VALUE="fin"&gt;Finance &lt;OPTION VALUE="dev"&gt;Development &lt;OPTION VALUE="prd"&gt; Production&lt;/SELECT&gt;&lt;/TD&gt; &lt;/TR&gt; &lt;/TABLE&gt; </pre>
--	--	--

### Let's check your progress:

5. Which field allows user to enter multiple line data?  
 a. text                      b. password                      c. textarea  
 d. hidden

---

## 6.8 PASSWORD

---

PASSWORD indicates that the field is for typing in a password. PASSWORD works just like a TEXT type field, with the difference that whatever is typed is not displayed on the screen (in case someone is watching over your shoulder or you have to leave the work station). Instead of showing what you typed in, the browser displays a series of asterisks (\*), bullets (·), or something to show that you are typing, but not what you are typing. So, for example, this code:

```

<FORM ACTION="..../cgi-bin/mycgi.pl" METHOD=POST>
name: <INPUT TYPE=TEXT NAME="realname"><BR>
password: <INPUT TYPE=PASSWORD NAME="mypassword">
<P><INPUT TYPE=SUBMIT VALUE="submit">
</FORM>

```

### Output:

The screenshot shows a web form with the following elements:

- A label "name:" followed by a text input field containing the text "Suresh".
- A label "password:" followed by a password input field containing seven black dots (•••••••).
- A "submit" button below the password field.

Note that PASSWORD fields are not sent encrypted, they are sent in the same manner as all the other elements on the form: in the clear text. Note also that when you use PASSWORD you should also set the form METHOD to POST.

---

## 6.9 FILE UPLOAD

---

FILE is used for doing file uploads in a form. File uploads are a relatively new and still not well-standardized type of form input, but they show great promise once the bugs are ironed out. File uploads allow you to send an entire file from your computer to the web server as part of your form input.

```
<FORM METHOD=POST ENCTYPE="multipart/form-data"
ACTION="./cgi-bin/mycgi.pl">
File to upload: <INPUT TYPE=FILE NAME="upfile"><BR>
<INPUT TYPE=SUBMIT VALUE="Submit">
</FORM>
```

The image shows a rendered HTML form. It consists of a text input field with the label 'File to upload:' to its left. To the right of the input field is a button labeled 'Browse...'. Below the input field and 'Browse...' button is another button labeled 'Submit'.

Configuring a form for file uploads requires setting two attributes in the <FORM> tags in addition to using <INPUT TYPE=FILE>: POST and "multipart/form-data" (as in the example above). When the data is sent, the original file name (including the full path) of the file as it was on your computer is sent to the web server. The Scripting code, however, is free to save the file as anything it wants -- or to not save it at all.

An often expressed wish with file uploads is to have a way of suggesting the file type being uploaded. Netscape, for example, when it gives you the file upload dialog box, inexplicably assumes you want to upload an HTML file. Unfortunately, there is no way to suggest a file type.

### Let's check your progress:

6. What should be the value of ENCTYPE if file upload field is used in web page?
- a. multipart/form-data      b. multipart      c.      text/form-data  
d. form-data

---

## 6.10 BUTTON

---

BUTTON defines a button which causes a script to run. Use the onClick attribute to give the script command(s). BUTTON is used only with scripting. Browsers that don't understand scripts don't understand this type of input and usually render it as a text input field.

```
<FORM>
<TABLE BORDER CELLPADDING=3>
<TR>
```

```

<TD><NOBR>radius: <INPUT NAME="Circle_radius"
SIZE=4></NOBR></TD>
<TD><INPUT TYPE=BUTTON
OnClick="Circle_calc(this.form);" VALUE="calculate"></TD>
<TD ALIGN=RIGHT BGCOLOR="#AACCFF">
<NOBR>circumference: <INPUT
NAME="Circle_circumference" SIZE=9></NOBR><BR>
<NOBR>area: <INPUT NAME="Circle_area"
SIZE=9></NOBR></TD>
</TR>
</TABLE>
</FORM>

```

**Output:**
**<BUTTON>:**

Another way to render button on a web page is using <BUTTON> tag. <BUTTON> creates a button. Unlike <INPUT>, <BUTTON> is a container which allows you to put regular HTML contents in the button, including text and pictures. Unfortunately, <BUTTON> does not degrade well, and so at this time it's best to stick with <INPUT>. <BUTTON> tag has following attributes:

TYPE: what type of button is this      DISABLED: disable this button  
onClick: script to run when the user      ACCESSKEY: shortcut key for this  
clicks here      button  
NAME: name of this button element      TABINDEX: tab order  
VALUE: the value sent with the form

```

<BUTTON TYPE=SUBMIT><IMG SRC="Penguins.jpg" HEIGHT=97
WIDTH=105 ALT="Starflower"
ALIGN="ABSMIDDLE"><STRONG>Send It
In!</STRONG></BUTTON>

```

**Output:**

By default, <BUTTON> creates a plain button, much like <INPUT TYPE=BUTTON>. With the TYPE attribute, <BUTTON> can also create submit and reset buttons. The HTML code put between <BUTTON> and </BUTTON> is not the value sent with the form. The value of the button determined by the <INPUT VALUE="..."> attribute.

**SUBMIT**

SUBMIT creates the "Submit" button which sends the form in to the CGI. In its simplest form, you can use SUBMIT and no other attributes for the <INPUT ...> tag:

```
<FORM ACTION="../cgi-bin/mycgi.pl">
name: <INPUT NAME="realname"><BR>
email: <INPUT NAME="email"><P>
<INPUT TYPE=SUBMIT>
</FORM>
```

**Output:**

name:

email:

You can customize the text used for the button using the VALUE attribute:

```
<FORM ACTION="../cgi-bin/mycgi.pl">
name: <INPUT NAME="realname"><BR>
email: <INPUT NAME="email"><P>
<INPUT TYPE=SUBMIT VALUE="Send It!">
</FORM>
```

**Output:**

name:

email:

You may sometimes find that you want to have more than one submit button on a form. If you give each button the same name, but different values, the browser will indicate which submit button was pressed:

```
<FORM ACTION="../cgi-bin/mycgi.pl">Go to the check-out
page?
<INPUT TYPE=SUBMIT NAME="checkout" VALUE="YES">
<INPUT TYPE=SUBMIT NAME="checkout" VALUE="NO">
</FORM>
```

**Output:**

Go to the check-out page?

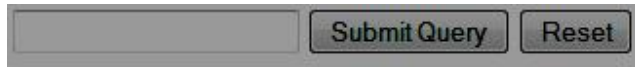
Let's check your progress:

7. Can we place more than one submit button in an html form?
- a. yes                      b. no

**RESET**

RESET resets the form so that it is the way it was before anything was typed in:

```
<FORM ACTION="./cgi-bin/mycgi.pl">
<INPUT TYPE=TEXT>
<INPUT TYPE=SUBMIT>
<INPUT TYPE=RESET>
</FORM>
```

**Output:**


If you do choose to use have a reset button in your form, consider adding a check if the user actually wants to reset. You can do this by adding an onReset event handler to the <FORM> tag:

```
<FORM ACTION="./cgi-bin/mycgi.pl"
  onReset="return confirm('Do you really want to reset the form?')">
<INPUT TYPE=TEXT NAME="query">
<INPUT TYPE=SUBMIT>
<INPUT TYPE=RESET>
</FORM>
```

---

## 6.11 RADIO

---

RADIO is used to create a series of choices of which only one can be selected. The term "radio button" comes from the buttons for the radio in an automobile, where selecting one radio station automatically de-selects all the others. HTML radio buttons are created by using several <INPUT TYPE=RADIO> buttons, all with the same name, but with different values. For example, this series of buttons allows you to choose one size for a pizza:

```
<FORM ACTION="./cgi-bin/mycgi.pl">
What size pizza?<P>
<INPUT TYPE=RADIO NAME="pizzasize" VALUE="S"
>small<BR>
<INPUT TYPE=RADIO NAME="pizzasize" VALUE="M"
CHECKED >medium<BR>
<INPUT TYPE=RADIO NAME="pizzasize" VALUE="L"
>large<P>
<INPUT TYPE=SUBMIT VALUE="submit">
</FORM>
```

**Output:**

What size pizza?

small  
 medium  
 large

If no CHECKED attribute is used, different browsers have different ways of displaying the initial state of a series of radio buttons. Netscape and MSIE have none of the buttons selected. Mosaic selects the first button.

---

## 6.12 CHECKBOX

---

CHECKBOX creates a checkbox which can be either on or off:

```
<FORM ACTION="../cgi-bin/mycgi.pl">
<INPUT TYPE=CHECKBOX NAME="mushrooms"
>mushrooms<BR>
<INPUT TYPE=CHECKBOX NAME="greenpeppers">green
peppers<BR>
<INPUT TYPE=CHECKBOX NAME="olives"    >olives<BR>
<INPUT TYPE=CHECKBOX NAME="onions"    >onions<P>
<INPUT TYPE=SUBMIT VALUE="submit">
</FORM>
```

**Output:**

mushrooms  
 green peppers  
 olives  
 onions

By default, the checkbox is initially off. If you want the checkbox initially on, use the CHECKED attribute. Checkbox CHECKBOXs are only sent to the scripting code if they are on; if they are off, no name/value pair is sent (try out the form above to see).



---

## 6.13 SELECT

---

<SELECT> sets up a list of choices from which a user can select one or many. <SELECT> tag takes following attributes:



Name	Description	Example
NAME	NAME names the select field for use with scripting. NAME works just like <code>&lt;INPUT NAME="..."&gt;</code>	
MULTIPLE	MULTIPLE designates that more than one option in the list can be selected. When creating a multiple list it is almost always a good idea to also use the SIZE attribute.	<pre data-bbox="842 533 1257 1106">&lt;SELECT NAME="toppings" MULTIPLE SIZE=5&gt; &lt;OPTION VALUE="mushrooms"&gt;mushro oms &lt;OPTION VALUE="greenpeppers"&gt;green peppers &lt;OPTION VALUE="onions"&gt;onions &lt;OPTION VALUE="tomatoes"&gt;tomatoes &lt;OPTION VALUE="olives"&gt;olives &lt;/SELECT&gt;</pre> 
SIZE	<p data-bbox="512 1451 703 1485">SIZE = integer</p> <p data-bbox="512 1541 820 1675">SIZE indicates how many rows of the list should be displayed. The default is one.</p>	<p data-bbox="842 1451 1257 1518">For example, the following code creates a select list with 6 rows</p> <pre data-bbox="842 1574 1209 1641">&lt;SELECT NAME="county" SIZE=6&gt;</pre> 

DISABLED	DISABLED works exactly the same as the corresponding attribute for <INPUT>.	
TABINDEX	<p>TABINDEX = integer</p> <p>Normally, when the user tabs from field to field in a form (in a browser that allows tabbing, not all browsers do) the tab order is the order the fields appear in the HTML code.</p> <p>However, sometimes you want the tab order to flow a little differently. In that case, you can number the fields using TABINDEX. The tabs then flow in order from lowest TABINDEX to highest.</p>	<pre>&lt;TABLE BORDER CELLPADDING=3 CELLSPACING=5 BGCOLOR="#FFFFCC"&gt; &lt;TR&gt;   &lt;TD&gt;name: &lt;INPUT NAME="realname" TABINDEX=1&gt;&lt;/TD&gt;   &lt;TD ROWSPAN=3&gt;comments&lt;BR&gt;   &lt;TEXTAREA COLS=25 ROWS=5 TABINDEX=4&gt;&lt;/TEXTAREA&gt; &lt;/TD&gt;&lt;/TR&gt; &lt;TR&gt; &lt;TD&gt;email: &lt;INPUT NAME="email" TABINDEX=2&gt;&lt;/TD&gt;&lt;/TR&gt; &lt;TR&gt; &lt;TD&gt;department: &lt;SELECT NAME="dep" TABINDEX=3&gt;   &lt;OPTION VALUE=""&gt;...   &lt;OPTION VALUE="mkt"&gt;Marketing   &lt;OPTION VALUE="fin"&gt;Finance   &lt;OPTION VALUE="dev"&gt;Development   &lt;OPTION VALUE="prd"&gt;Production&lt;/SE LECT&gt;&lt;/TD&gt;&lt;/TR&gt; &lt;/TABLE&gt;</pre>

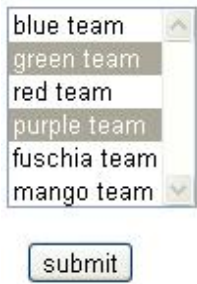
---

## 6.14 OPTION

---

<OPTION ...> is used along with <SELECT> to create select lists. <OPTION> indicates the start of a new option in the list. <OPTION> can be used without any attributes, but you usually need the VALUE attribute, which indicates what is sent to the server. The text which follows <OPTION> is what is displayed in the browser. <OPTION> takes following attributes:

Name	Description	Example
VALUE	VALUE indicates the value that is sent to the server if that option is chosen. The value of VALUE is not seen by the user.	<pre data-bbox="821 365 1267 757">&lt;SELECT NAME="partnumber"&gt; &lt;OPTION VALUE="7382"&gt;steam turbine &lt;OPTION VALUE="2928"&gt;resistor array &lt;OPTION VALUE="3993"&gt;widget analyzer &lt;OPTION VALUE="9398"&gt;fiber identifier &lt;/SELECT&gt;</pre> <p data-bbox="821 813 935 846"><b>Output:</b></p> <div data-bbox="831 860 1062 972"> <input data-bbox="831 860 1062 898" type="text" value="steam turbine"/> <input data-bbox="852 931 959 972" type="button" value="submit"/> </div> <p data-bbox="821 1037 1267 1211">In this example, if you selected the first option, "steam turbine", then the name/value pair partnumber=7382 is sent to the scripting code.</p>
SELECTED	SELECTED indicates that the option should be selected by default.	<p data-bbox="821 1234 1246 1335">In this example, the third item ("widget analyzer") is the default item:</p> <pre data-bbox="821 1395 1267 1787">&lt;SELECT NAME="partnumber"&gt; &lt;OPTION VALUE="7382" &gt;steam turbine &lt;OPTION VALUE="2928" &gt;resistor array &lt;OPTION VALUE="3993" SELECTED &gt;widget analyzer &lt;OPTION VALUE="9398" &gt;fiber identifier &lt;/SELECT&gt;</pre> <p data-bbox="821 1843 935 1877"><b>Output:</b></p> <div data-bbox="831 1890 1062 2002"> <input data-bbox="831 1890 1062 1928" type="text" value="widget analyzer"/> <input data-bbox="852 1962 959 2002" type="button" value="submit"/> </div>

		<p>SELECTED can also be used in multiple select lists. In this example, the "green team" and the "purple team" are the default selected items:</p> <pre>&lt;SELECT NAME="teams" MULTIPLE SIZE=6&gt; &lt;OPTION VALUE="b" &gt;blue team &lt;OPTION VALUE="g" SELECTED &gt;green team &lt;OPTION VALUE="r"      &gt;red team &lt;OPTION VALUE="p" SELECTED &gt;purple team &lt;OPTION VALUE="f" &gt;fuschia team &lt;OPTION VALUE="m" &gt;mango team &lt;/SELECT&gt;</pre> <p><b>Output:</b></p> 
--	--	---

### Let's check your progress:

8. If you want to a drop down box with five cities name in it. How many <OPTION> tags are required?
- a. 2                      b.3                      c.5                      d.1

---

## 6.15 FORMS AND SCRIPTING

---

### Source code:

```
<SCRIPT TYPE="text/javascript">
<!--
function Circle_calc(GeoForm)
{
var CircleRadius = GeoForm.Circle_radius.value;
```

```

if (CircleRadius >= 0)
{
    GeoForm.Circle_circumference.value = 2 * Math.PI * CircleRadius ;
    GeoForm.Circle_area.value = Math.PI * Math.pow(CircleRadius, 2) ;
}
else
{
    GeoForm.Circle_circumference.value = "";
    GeoForm.Circle_area.value = "";
}
}
function Cone_calc(GeoForm)
{
    var ConeRadius = GeoForm.Cone_radius.value;
    var ConeHeight = GeoForm.Cone_height.value;
    if ((ConeRadius >= 0) && (ConeHeight >= 0))
    {
        GeoForm.Cone_surfacearea.value = (Math.PI * Math.pow(ConeRadius,
2)) +
(Math.PI * ConeRadius * Math.sqrt(Math.pow(ConeRadius, 2) +
Math.pow(ConeHeight, 2)));
        GeoForm.Cone_volume.value = (1/3) * Math.PI *
Math.pow(ConeRadius,2) *
ConeHeight;
    }
    else
    {
        GeoForm.Cone_surfacearea.value = "";
        GeoForm.Cone_volume.value = "";
    }
}
function Sphere_calc(GeoForm)
{
    var SphereRadius = GeoForm.Sphere_radius.value;
    if (SphereRadius >= 0)
    {
        GeoForm.Sphere_surfacearea.value = 4 * Math.PI *
Math.pow(SphereRadius, 2);
        GeoForm.Sphere_volume.value = (4/3) * Math.PI *
Math.pow(SphereRadius, 3);
    }
}

```

```

    }
else
    {
        GeoForm.Sphere_surfacearea.value = "";
        GeoForm.Sphere_volume.value = "";
    }
}
// -->
</SCRIPT>
<FORM>
<TABLE BORDER CELLPADDING=3>
<!-- circumference and radius of a circle -->
<TR>
<TH>Circumference and Radius of a Circle<BR><IMG
SRC=" ../graphics/circle.gif"
HEIGHT=88 WIDTH=88 ALT="picture of a circle"></TH>
<TD><NOBR>radius: <INPUT NAME="Circle_radius"
SIZE=4></NOBR></TD>
<TD><INPUT TYPE=BUTTON onClick="Circle_calc(this.form);"
VALUE="calculate"></TD>
<TD ALIGN=RIGHT BGCOLOR="#AACCFF">
    <NOBR>circumference: <INPUT NAME="Circle_circumference"
SIZE=9></NOBR><BR>
    <NOBR>area: <INPUT NAME="Circle_area"
SIZE=9></NOBR></TD>
</TR>
<!-- Surface area and volume of a cone -->
<TR>
<TH>Surface Area and Volume of a Cone<BR><IMG
SRC=" ../graphics/cone.gif" HEIGHT=106 WIDTH=115 ALT="picture of
a cone"></TH>
<TD ALIGN=RIGHT><NOBR>radius: <INPUT NAME="Cone_radius"
SIZE=4></NOBR><BR>
    <NOBR>height: <INPUT NAME="Cone_height"
SIZE=4></NOBR></TD>
<TD><INPUT TYPE=BUTTON onClick="Cone_calc(this.form);"
VALUE="calculate"></TD>
<TD ALIGN=RIGHT BGCOLOR="#AACCFF">
    <NOBR>surface area: <INPUT NAME="Cone_surfacearea"
SIZE=9></NOBR><BR>

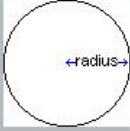
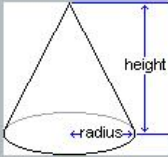
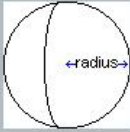
```

```

    <NOBR>volume: <INPUT NAME="Cone_volume"
SIZE=9></NOBR></TD>
</TR>
<!-- Surface Area and Volume of a Sphere -->
<TR>
<TH>Surface Area and Volume of a Sphere<BR><IMG
SRC=" ../graphics/sphere.gif" HEIGHT=88 WIDTH=88 ALT="picture of
a sphere"></TH>
<TD><NOBR>radius: <INPUT NAME="Sphere_radius"
SIZE=4></NOBR></TD>
<TD><INPUT TYPE=BUTTON onClick="Sphere_calc(this.form);"
VALUE="calculate"></TD>
<TD ALIGN=RIGHT BGCOLOR="#AACCFF"><NOBR>surface area:
<INPUT NAME="Sphere_surfacearea" SIZE=9></NOBR><BR>
    <NOBR>volume: <INPUT NAME="Sphere_volume"
SIZE=9></NOBR></TD>
</TR>
</TABLE>
</FORM>

```

**Output:**

<b>Circumference and Radius of a Circle</b> 	radius: <input type="text"/>	<input type="button" value="calculate"/>	circumference: <input type="text"/> area: <input type="text"/>
<b>Surface Area and Volume of a Cone</b> 	radius: <input type="text"/> height: <input type="text"/>	<input type="button" value="calculate"/>	surface area: <input type="text"/> volume: <input type="text"/>
<b>Surface Area and Volume of a Sphere</b> 	radius: <input type="text"/>	<input type="button" value="calculate"/>	surface area: <input type="text"/> volume: <input type="text"/>

**Explanation:**

In the above code three functions are defined in <SCRIPT> tag namely Circle\_calc(GeoForm), Cone\_calc(GeoForm), Sphere\_calc(GeoForm).

Circle\_calc(GeoForm) function calculates circumference and area of a circle based on the value of radius entered by user( value of radius is stored using the code: var CircleRadius = GeoForm.Circle\_radius.value;).

This value of radius is used in formula to calculate circumference and area and displayed in respective text fields.

Similarly `Cone_calc(GeoForm)`, `Sphere_calc(GeoForm)` are defined for cone and sphere.

These functions are called in the HTML code when user clicks on calculate button for circle, cone and sphere (i.e. onclick event of calculate button).

---

## 6.16 LABELING INPUT FILES

---

`<LABEL ...>`, an HTML 4.0 element supported by MSIE and Netscape 6, defines a set of text that is associated with a particular form element.

`<LABEL>` tag takes one attribute i.e. FOR:  
FOR = "text string"

If the form element that should be associated with the text in `<LABEL>` can't be contained within `<LABEL>`, such as when the form element in another table cell, use FOR. The value of FOR is the ID of the form element. Note that it is not the name of the field (as given with the NAME attribute) it is the ID as given with the ID attribute. So, for example, this code associates the text "join mailing list?" with the checkbox field that has the ID "joinlist":

```
<TABLE BORDER=1 CELLPADDING=5>
<TR> <TD><LABEL FOR="joinlist">join mailing
list?</LABEL></TD>
  <TD><INPUT TYPE=CHECKBOX NAME="joinlist"
ID="joinlist"></TD>
</TR>
</TABLE>
```

**Output:**

join mailing list?	<input type="checkbox"/>
--------------------	--------------------------

---

## 6.17 GROUPING RELATED FIELDS

---

**`<FIELDSET>`:**

`<FIELDSET>` defines a group of form elements as being logically related. The browser draws a box around the set of fields to indicate that they are related. For example, a form might contain a few fields about name and email, some fields asking for opinions, and a field for "other comments". `<FIELDSET>` could be used to group those fields like this:



```

<FIELDSET>
name: <INPUT NAME="realname"><BR>
email: <INPUT NAME="email">
</FIELDSET><P>

```

```

<FIELDSET>
favorite color: <INPUT NAME="favecolor"><BR>
<INPUT TYPE="CHECKBOX" NAME="onions"> like green
onions<BR>
<INPUT TYPE="CHECKBOX" NAME="cookies"> like
cookies<BR>
<INPUT TYPE="CHECKBOX" NAME="kimchee"> like kim
chee<BR>
</FIELDSET><P>

```

```

<FIELDSET>
other comments:<BR>
<TEXTAREA NAME="comments" ROWS=5
COLS=25></TEXTAREA>
</FIELDSET>

```

**Output:**

name: <input type="text"/> email: <input type="text"/>
favorite color: <input type="text"/> <input type="checkbox"/> like green onions <input type="checkbox"/> like cookies <input type="checkbox"/> like kim chee
other comments: <div style="border: 1px solid black; height: 50px; width: 150px;"></div>

**<LEGEND>:**

<LEGEND> is used with <FIELDSET> to give a title to each set of fields. Attribute of <LEGEND> is ALIGN. ALIGN align the <LEGEND> text left, center, or right

```

<FIELDSET>
<LEGEND ALIGN=LEFT>Personal Stuff</LEGEND><P>
name: <INPUT NAME="realname"><BR>

```

email: <INPUT NAME="email">  
</FIELDSET><P>

<FIELDSET>  
<LEGEND ALIGN=CENTER>Survey</LEGEND><P>  
favorite color: <INPUT NAME="favecolor"><BR>  
<INPUT TYPE=CHECKBOX NAME="favecolor"> like green  
onions<BR>  
<INPUT TYPE=CHECKBOX NAME="onions"> like  
cookies<BR>  
<INPUT TYPE=CHECKBOX NAME="kimchee"> like kim  
chee<BR>  
</FIELDSET><P>

<FIELDSET>  
<LEGEND ALIGN=RIGHT>Misc</LEGEND><P>  
other comments:<BR>  
<TEXTAREA NAME="comments" ROWS=5  
COLS=25></TEXTAREA>  
</FIELDSET>

**Output:**

Personal Stuff
name: <input type="text"/>
email: <input type="text"/>

Survey
favorite color: <input type="text"/>
<input type="checkbox"/> like green onions
<input type="checkbox"/> like cookies
<input type="checkbox"/> like kim chee

Misc
other comments: <input type="text"/>

---

## 6.18 FORM FIELD EVENT HANDLERS

---

Following are the event handlers which can be used with form fields:

### onClick:

onClick gives the script to run when the user clicks on the input. onClick applies to buttons (submit, reset, and button), checkboxes, radio buttons, and form upload buttons. onClick is mostly used with plain button type inputs:

```
<FORM>
<TABLE BORDER CELLPADDING=3>
<TR>
<TD><NOBR>radius: <INPUT NAME="Circle_radius"
SIZE=4></NOBR></TD>
<TD><INPUT TYPE=BUTTON onClick="Circle_calc(this.form);"
VALUE="calculate"></TD>
<TD ALIGN=RIGHT BGCOLOR="#AACCFF">
<NOBR>circumference: <INPUT NAME="Circle_circumference"
SIZE=9></NOBR><BR>
<NOBR>area: <INPUT NAME="Circle_area" SIZE=9></NOBR></TD>
</TR>
</TABLE>
</FORM>
```

onClick can return a value to possibly cancel the action the button would normally perform. For example, we can use onClick to double check if the user really wants to reset the form data. In this form, if you click on the reset button, you get a dialog box asking if you really want to cancel. If you choose "cancel", the form is not reset.

```
<INPUT TYPE=RESET onClick="return confirm('Are you sure you
want to reset the form?')" >
```

### OnChange:

onChange specifies script code to run when the data in the input field changes. onChange applies to input fields which accept text, namely text and password fields. (<TEXTAREA> and <SELECT> fields also use onChange.)

The onChange event is triggered when the contents of the field changes. For example, when the user enters an email address in this form, a script does some basic validity checking on the value entered:

```
<SCRIPT TYPE="text/javascript">
<!--
function checkEmail(email)
{
```

```

if(email.length > 0)
{
  if (email.indexOf(' ') >= 0)
    alert("email addresses cannot have spaces in them");
  else if (email.indexOf('@') == -1)
    alert("a valid email address must have an @ in it");
  }
}
//-->
</SCRIPT>
<FORM ACTION="../cgi-bin/mycgi.pl" METHOD=POST>
name:   <INPUT NAME="realname"><BR>
email:  <INPUT NAME="email"
onChange="checkEmail(this.value)"><BR>
destination: <INPUT NAME="desination">
</FORM>

```

Because `onChange` only occurs when the value changes, the user is only warned once about a bad value. That's generally the most desirable way to do error checking. Even if the value is wrong, most users prefer to be warned just once.

### **onFocus and onBlur:**

`onFocus` is the event handler for when the input field gets the focus. `onBlur` is the event handler for when the input field loses the focus. The "focus" indicates which object in the window reacts to keyboard input. If a text field has the focus, then if you type something in the keyboard, the letters appear in that field. Focus shifts from one object to another usually by clicking on them with the mouse or by hitting the tab key.

An example of `onFocus` and `onBlur` used to change the status bar. Notice that we use `onFocus` to give the message we want, and `onBlur` to change back to the default.

```

<FORM ACTION="../cgi-bin/mycgi.pl" METHOD=POST>
name: <INPUT
      NAME="realname"
      onFocus = "window.status='Enter your name'"
      onBlur = "window.status=window.defaultStatus"
      ><BR>
email: <INPUT
      NAME="email"
      onFocus = "window.status='Enter your email address'"

```

```

        onBlur = "window.status=window.defaultStatus"
    >
<P><INPUT TYPE=SUBMIT VALUE="submit">
</FORM>

```

**onKeyPress:**

```

<HTML>
<HEAD>
<SCRIPT TYPE="text/javascript">
<!--
function numbersonly(myfield, e, dec)
{
var key;
var keychar;
if (window.event)
    key = window.event.keyCode;
else if (e)
    key = e.which;
else
    return true;
keychar = String.fromCharCode(key);
// control keys
if ((key==null) || (key==0) || (key==8) ||
    (key==9) || (key==13) || (key==27) )
    return true;
// numbers
else if ((("0123456789").indexOf(keychar) > -1))
    return true;
// decimal point jump
else if (dec && (keychar == "."))
    {
    myfield.form.elements[dec].focus();
    return false;
    }
else
    return false;
}
//-->
</SCRIPT>
</HEAD>
<BODY>

```

```

<FORM ACTION="../cgi-bin/mycgi.pl" METHOD=POST>
U.S. ZIP Code:
  <INPUT NAME="dollar" SIZE=5 MAXLENGTH=5
    onKeyPress="return numbersonly(this, event)">
  <INPUT TYPE=SUBMIT VALUE="go">
</FORM>
</BODY>
</HTML>

```

In the above code whenever a user enters any value through keyboard (i.e. typing any key), onKeyPress event handler is called. When this event handler is triggered function numbersonly() is called which allows only numbers to be entered.

---

## 6.19 PASSING FORM DATA

---

Whenever we are creating HTML forms, our objective is to retrieve the values entered by the user. These values are called form data. This form data can be passed to another page using method and action attributes (discussed in <FORM> tag) of form tag. The page name specified in form's action attribute is the one to which form data is passed, and then processed by this page.

```

<HTML>
<BODY>
<FORM ACTION="insert.php" METHOD="post">
Firstname: <INPUT TYPE="text" NAME="firstname" />
Lastname: <INPUT TYPE="text" NAME="lastname" />
Age: <INPUT TYPE="text" NAME="age" />
<INPUT TYPE="submit" />
</FORM>
</BODY>
</HTML>

```

In the above form user will enter first name and last name in text boxes. When form is submitted insert.php page is called and form data (i.e. first name and last name) is passed to this page. Passing data from insert.html to insert.php is done through post method. Once the form data is with insert.php, it can process the data.

---

## 6.20 SUMMARY

---

In this chapter we discussed role of form tag, different elements (input, select, textarea, label, option, etc.) and their attributes that can go within form tags. We also discussed how to use event handlers in form.

**Answers to check your progress**

1.b            2. d            3. a            4.b            5.c  
6.a            7.a            8.c

---

**6.21 EXERCISE**

---

**6.21.1 Questions**

1. Explain <FORM> tag and its attributes
2. Explain text field, password field, hidden field form elements and how they can be placed on web page.
3. How to put Drop down box in a web page?
4. Explain <fieldset> and <legend> tags in detail.
5. Explain with any four event handlers that can be used with form fields.
6. Explain how to place radio buttons and check boxes on web page.

**6.21.2 Programs**

1. Design a feedback form that will have following fields:  
Faculty's name (text field), year (text field), Is professor explaining properly: yes, no (radio button), Any Comments (text area), submit button.
2. Design a registration form that will have following fields:  
User name, password, address, phone no., Gender (male, female), emailed, and a submit button. (Use appropriate form elements wherever necessary)
3. Design a form that will have following fields:  
A text field to enter a string, four radio buttons for colors (red, green, blue, yellow), and a submit button. Whenever user enters a string, selects a color and submits the form the string should appear in that color.



# JAVASCRIPT

## Unit Structure

- 7.1 Objective
- 7.2 Introduction
  - 7.2.1 Client side javascript
  - 7.2.2 Server side javascript
  - 7.2.3 Javascript objects
  - 7.2.4 Javascript security
- 7.3 Operators
  - 7.3.1 Assignment
  - 7.3.2 Arithmetic
  - 7.3.3 Comparison
  - 7.3.4 Modulus
  - 7.3.5 Increment
  - 7.3.6 Decrement
  - 7.3.7 Unary negation
  - 7.3.8 Logical operators
  - 7.3.9 Short circuit evaluation
  - 7.3.10 String operators
  - 7.3.11 Special operators
  - 7.3.12 Conditional operators
  - 7.3.13 Comma operators
  - 7.3.14 Delete
  - 7.3.15 New
  - 7.3.16 This
  - 7.3.17 Void
- 7.4 Statements
  - 7.4.1 Break
  - 7.4.2 Comment
  - 7.4.3 Continue
  - 7.4.4 Delete
  - 7.4.5 Do while
  - 7.4.6 Export
  - 7.4.7 For
  - 7.4.8 For in



- 7.4.9 Function
- 7.4.10 If else
- 7.4.11 Import
- 7.4.12 Labelled
- 7.4.13 Return
- 7.4.14 Switch
- 7.4.15 Var
- 7.4.16 While
- 7.4.17 With
- 7.5 Core java script
  - 7.5.1 Array
  - 7.5.2 Boolean
  - 7.5.6 Object
    - 7.5.6.1 Date
    - 7.5.6.2 Math
  - 7.5.7 Number
  - 7.5.8 Object
  - 7.5.9 String
  - 7.5.10 regExp

---

## 7.0 OBJECTIVE

---

After reading this chapter you will be able to –

- Write small javascripts with operators and functions .
- Use of variables, datatypes, control statements.
- Identify client and server side scripts
- Use javascript objects and their inbuilt properties.

---

## 7.1 INTRODUCTION

---

- Javascript is object based scripting language based on c++. Scripting language is a light weight programming language which easy to learn and understand. It is generally used for small applicatons.
- Javascript was basically designed to add interactivity in HTML pages and is directly embedded into HTML.
- Javascript is free to use by anyone.
- Javascript is interpreted language ie is not precompiled before execution. As javascript is interpreted, it is platform independent.

- Java is a full fledged complex programming language developed by sun microsystem. Javascript is developed by netscape communications and is no sub language of java.
- Javascript is originally a scripting language developed by European Computer Manufacturer's association (ECMA)
- Javascript that runs at the client side (ie at the client's browser) is client side java script (CCJS) and javascript that runs at the server is serverside java script (SSJS)
- Javascript being object oriented, uses number of built in javascript as well as objects can be created.
- Every object has properties and methods. Property is value(s) associated with an object. Methods are actions associated with an object.
- Example: 

```
<scripttype="text/javascript">
    document.write("This message is written by JavaScript");
</script>
```

in above example, 'document' is object and write() is a method of document object.
- Javascript runs in a web browser, and when a script written by a third party is executed on the browser, there is a risk of running a spyware or a virus program.
- Hence, each time javascript is loaded on the browser implements a security policy designed to minimise the risk of such unknown code.
- Security policy is set of rules governing what scripts can do under which circumstances.
- Modern javascript security is based upon Java. Scripts downloaded are isolated from the operating system and then executed. This is known as the 'sandbox' model. Some scripts are often stored randomly here and there. And hence, many times obtain more power than expected by design or by accident.
- Scripts in general are given limited access and more access is only given with the user consent. Taking a consent for every execution is not a practical solution.
- Scripts from 'trusted' source are many times excluded from this consent procedure.
- A policy called 'same origin' does not block scripts coming from the same origin as trusted scripts. This same origin check is performed on all methods of windows object, also on embedded and externally linked objects.

---

### 7.3 JAVASCRIPT OPERATORS

---

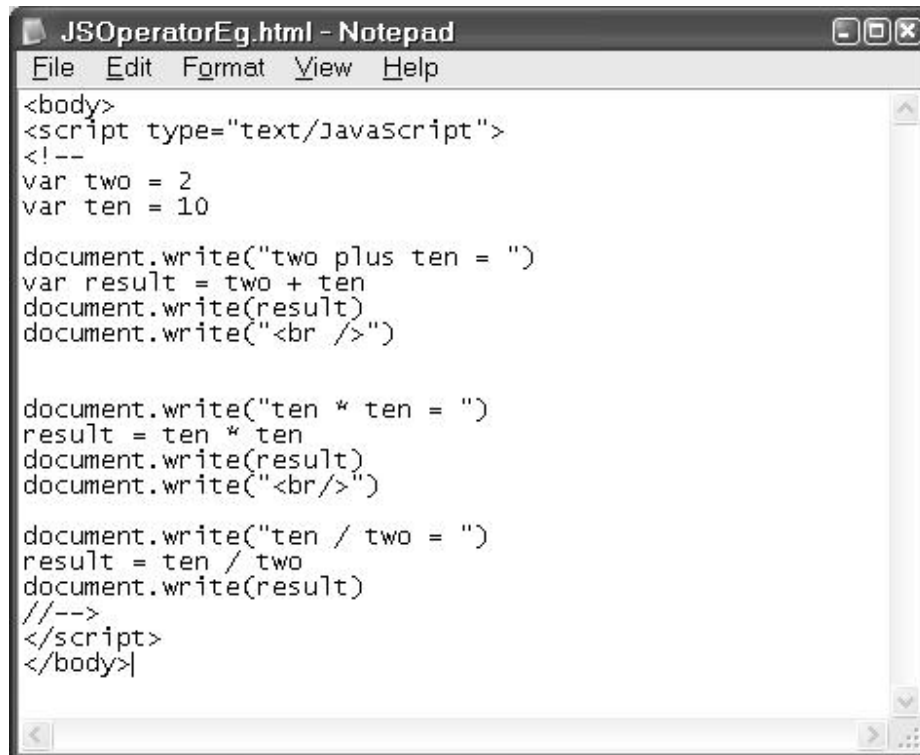
Operator	Name	Description	Example	Result	Note
=	Assignment	Assigns value to a variable.	y = 5; x = y = 5;	y = 5 x=5 and y=5	Primitive data types get direct values where as functions are pointers to the variables.
+=	Plus equal to		x += y	x =10	Same as operator equal to. That is, x+=y means that x=x+y and so on.
-=	Minus equal to		x -= y	x = 0	
*=	Multiply equal to		x *= y	x = 25	
/=	Divide equal to		x /= y	x =1	
%=	Modulo equal to		x %= y	x = 0	
Arithmetic					
+	Addition	Adds values	x = y + 2;	x=7	
-	Subtraction	Subtracts second from first value	x = y - 2;	x=3	
*	Multiplication	Multiplies two values	x = y * 2;	x=10	
/	Division	Divides	x = y / 2;	x=2.5	
%	Modulus	Divides and gives the remainder	x = y % 2;	x=1	
++	Increment	Increments value by 1	x = ++y;	x=6	
--	Decrement	Decrements value by 1	x = --y;	x=4	
Logical					
&&	And	Logically ands	(x<10 && y>1)	TRUE	As value of x as of now is 4 and that of y is 5
	Or	Logically Ors	(x==5    y==5)	TRUE	As value of x <> 5 but that of y = 5

!	Not	Negation	!(x==y)	TRUE	As x and y are not equal.
String					
+	Concatenation	Connects two strings.	If x = "5" and y= "5" then	x+y="55"	
Conditional					
?	Variable =(condition)?value1:value2		If a = 0, b = 7 a=(b=10)?5:8	a = 8, b = 7	If condition is satisfied, first value is assigned, else the second value is assigned.
Short circuit evaluation					
&&	And	Short circuit and	If x = 0; p = 66 x=p && p.getvalue()	x = 66	If p is not null, then assign value of p to x.
	SOr	Short circuit or	If x=0, default=5 x=default  10	x=5	If there is a default, assign default; else assign the given value.
,	Comma		x=4, y= 5;	x=4 y=5	Executes all expressions from left to right.
Delete		Used in functions	Delete <object>	Object undefined	Deletes the properties from objects and array elements from arrays making them undefined.
This	Can be used in following contexts – <ul style="list-style-type: none"> <li>• As global</li> <li>• Function context Simple call As object method Prototype chain</li> <li>• As a constructor</li> <li>• As getter or setter.</li> </ul>		this.window  return this; return this.value		
New		Creates new instance of the object	var x= new Fn()	x becomes new instance of function Fn and gets all its properties and methods.	

Void		Evaluates expression and returns undefined	Void expr	Undefined.	
------	--	--	-----------	------------	--

### Javascript Operators Examples

1.



```

JSOperatorEg.html - Notepad
File Edit Format View Help
<body>
<script type="text/JavaScript">
<!--
var two = 2
var ten = 10

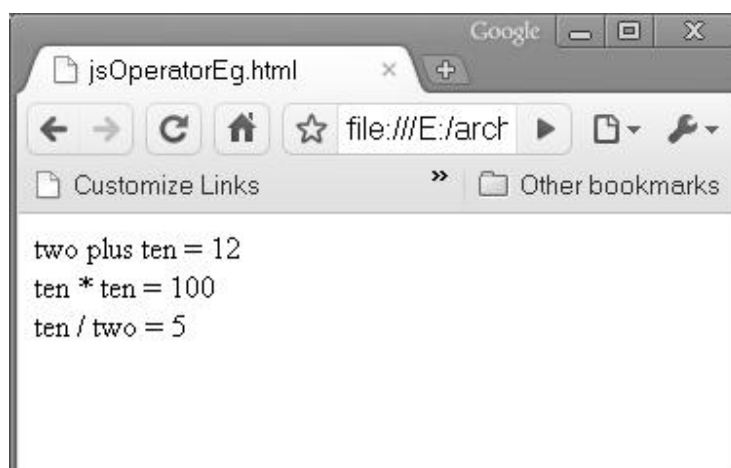
document.write("two plus ten = ")
var result = two + ten
document.write(result)
document.write("<br />")

document.write("ten * ten = ")
result = ten * ten
document.write(result)
document.write("<br />")

document.write("ten / two = ")
result = ten / two
document.write(result)
//-->
</script>
</body>

```

### Output



```

Google
jsOperatorEg.html
file:///E:/arch
two plus ten = 12
ten * ten = 100
ten / two = 5

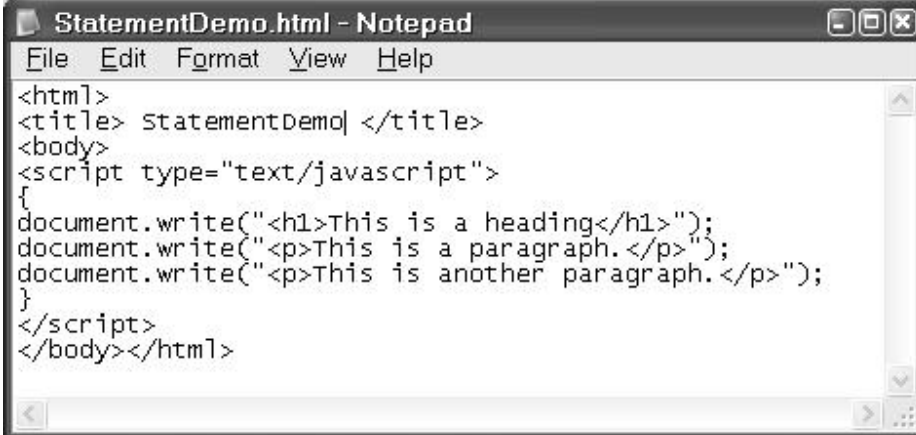
```

---

## 7.4 JAVASCRIPT STATEMENTS

---

- JavaScript is a sequence of statements to be executed by the browser. Browser executes the statements in the same order as they are written.
- JavaScript is case sensitive with all syntax, variable and function names.
- The semicolon at the end of line is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. However, semicolon at the end of line is good programming practice. Also, it enables us to write multiple statements on the same line.
- JavaScript statements can be grouped together in blocks. Blocks start with a left curly bracket {, and end with a right curly bracket }. The purpose of a block is to make the sequence of statements execute together.
- A block is normally used to group the statements in a function or condition.
- A general example of block is –



```
StatementDemo.html - Notepad
File Edit Format View Help
<html>
<title> statementDemo </title>
<body>
<script type="text/javascript">
{
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
}
</script>
</body></html>
```



### JavaScript Comments

- JavaScript comments can be used to make the code more readable.
- Comments can be added to explain the JavaScript.
- Comments can be added at end of a line.
- Single line comments start with //.
- Multi line comments start with /\* and end with \*/.
- The comment is used to prevent the execution of a single code line or a code block. This can be suitable for debugging
- Following example demonstrates use of comments in javascript code.

```

CommentsDemo.html - Notepad
File Edit Format View Help
<html>
<title> CommentsDemo </title>
<body>
<script type="text/javascript">
{
/* Following block is demonstration
of use of comments in javascript statements*/

document.write("<h1>This is a heading</h1>"); //First Heading
/*
Do not display second heading
document.write("<h2>This is First heading</h2>");
*/
document.write("<h3>This is third heading</h3>");//Third He.
document.write("<p>This is a paragraph.</p>");
//document.write("<p>This is another paragraph.</p>");
// Another paragraph not displayed
}
</script>
</body>
</html>

```

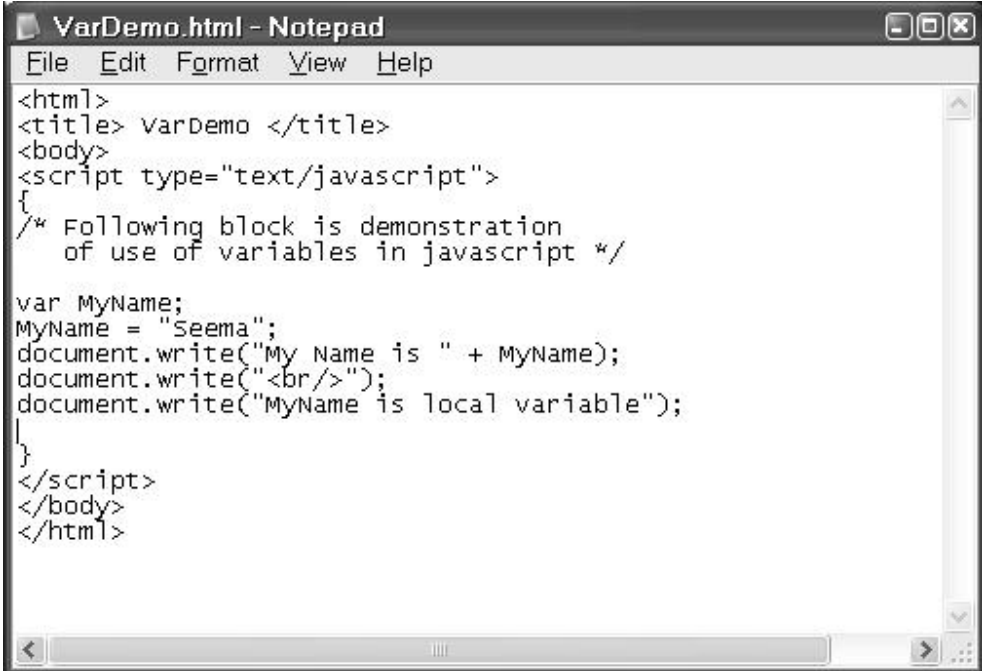


### JavaScript Variables (Var statement)

- Variables are "containers" for storing information. This information can be values or expressions.
- A variable can have a short name, like x, or a more descriptive name, like MyName.
- Rules for JavaScript variable names:
  - Variable names are case sensitive (y and Y are two different variables)
  - Variable names must begin with a letter or the underscore character
- Creating variables in JavaScript is most often referred to as "declaring" variables.
- You declare JavaScript variables with the var keyword like var x;
- After the declaration shown above, the variables are empty (they have no values yet). However, you can also assign values to the variables when you declare them like var x = 10; After the execution of this statement, the variable x will hold the value 10
- A variable declared within a JavaScript function becomes **LOCAL** and can only be accessed within that function. (the variable has local scope).
- You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
- Local variables are destroyed when you exit the function.
- Variables declared outside a function become **GLOBAL**, and all scripts and functions on the web page can access it.



- Global variables are destroyed when you close the page.
- If you declare a variable, without using "var", the variable always becomes GLOBAL.
- If you assign values to variables that have not yet been declared, the variables will automatically be declared as global variables.
- All javascript operators can be used with variables
- Example –



```
File Edit Format View Help
<html>
<title> varDemo </title>
<body>
<script type="text/javascript">
{
/* Following block is demonstration
of use of variables in javascript */

var MyName;
MyName = "Seema";
document.write("My Name is " + MyName);
document.write("<br/>");
document.write("MyName is local variable");
}
</script>
</body>
</html>
```



### Conditional statements

- Conditional statements are used to perform different actions based on different conditions.

- In JavaScript we have the following conditional statements:

- **If statement** - This statement is used to execute some code only if a specified condition is true.

- Syntax:

```

if(condition)
{
    code to be executed if condition is
    true
}

```

- **If...else statement** - This statement is used to execute some code if the condition is true and another code if the condition is false

- Syntax:

```

if (condition)
{
    code to be executed if condition is
    true
}
else
{
    code to be executed if condition is
    not true
}

```

- **If...else if...else statement** - This statement is used to select one of many blocks of code to be executed

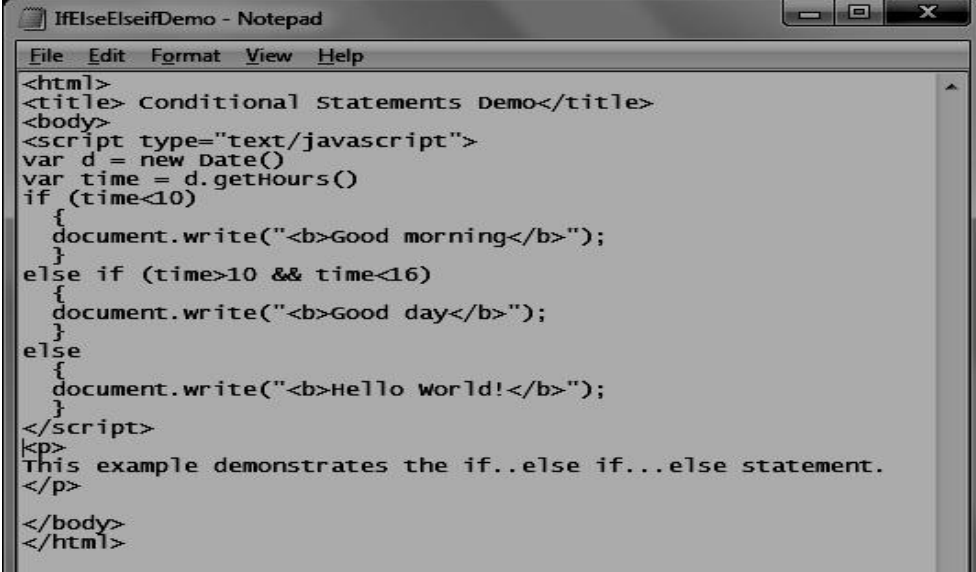
- Syntax:

```

if (condition1)
{
    code to be executed if condition1 is
    true
}
else if (condition2)
{
    code to be executed if condition2 is
    true
}
else
{
    code to be executed if neither
    condition1 nor condition2 is true
}

```

### Example



```

<html>
<title> Conditional Statements Demo</title>
<body>
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
{
document.write("<b>Good morning</b>");
}
else if (time>10 && time<16)
{
document.write("<b>Good day</b>");
}
else
{
document.write("<b>Hello world!</b>");
}
</script>
<p>
This example demonstrates the if..else if...else statement.
</p>
</body>
</html>

```

Output

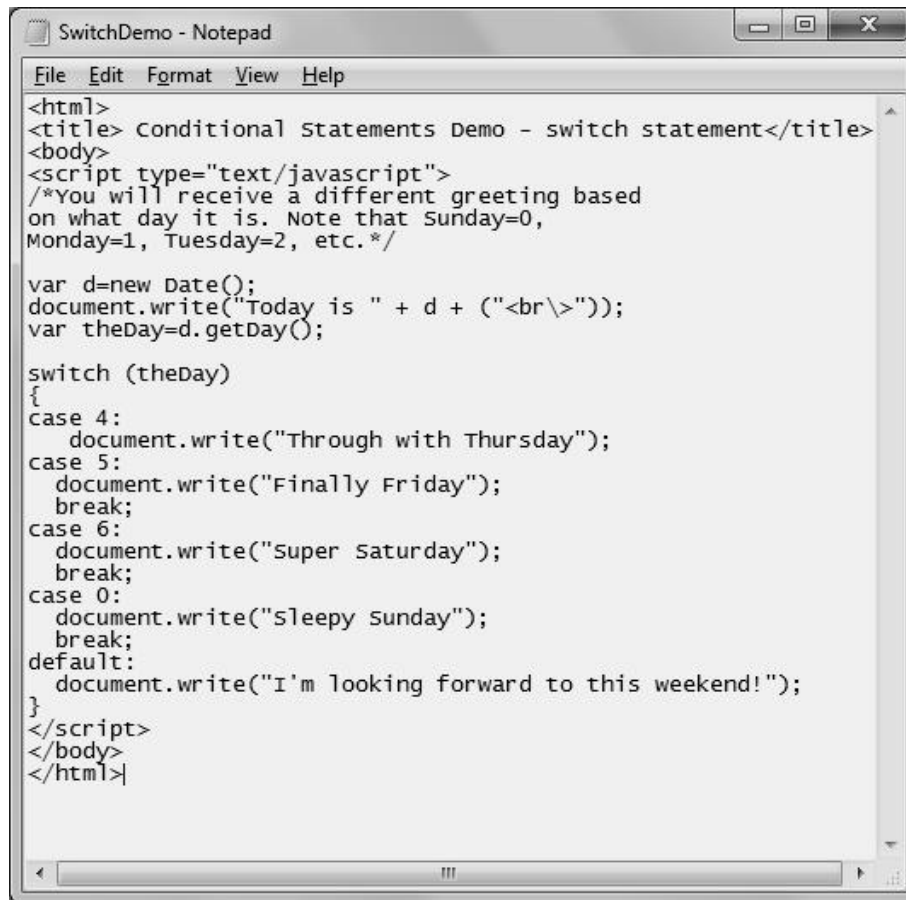


- **switch statement** -This statement is another way to select one of many blocks of code to be executed.
  - Syntax -

```

switch(n)
{
case 1:
execute code block 1
break;
case 2:
execute code block 2
break;
default:
code to be executed if n is different
from case 1 and 2
}

```

**Example:**


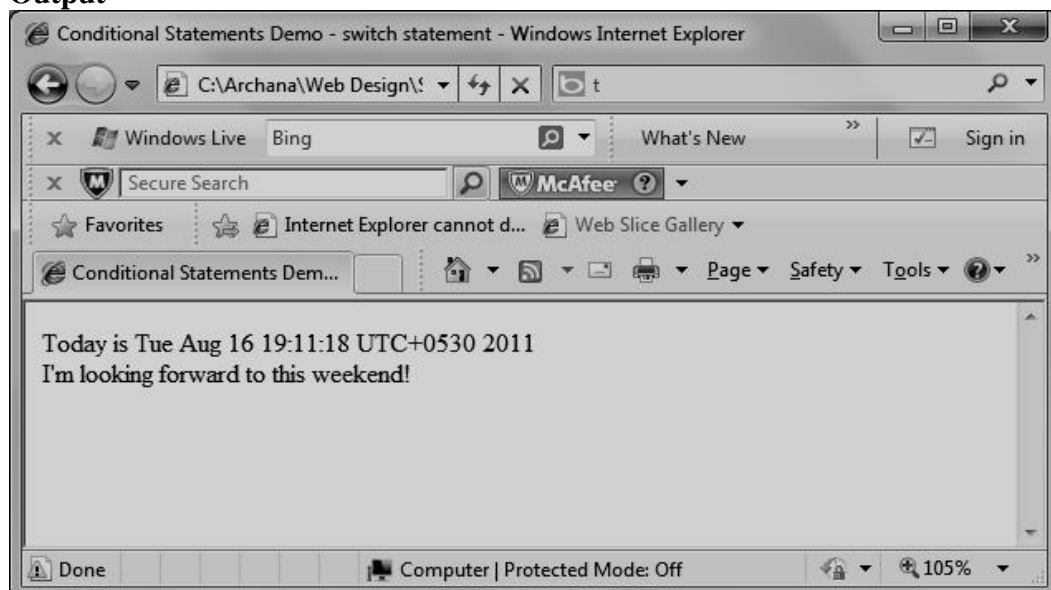
```

SwitchDemo - Notepad
File Edit Format View Help
<html>
<title> Conditional statements Demo - switch statement</title>
<body>
<script type="text/javascript">
/*You will receive a different greeting based
on what day it is. Note that Sunday=0,
Monday=1, Tuesday=2, etc.*/

var d=new Date();
document.write("Today is " + d + ("<br>"));
var theDay=d.getDay();

switch (theDay)
{
case 4:
    document.write("Through with Thursday");
case 5:
    document.write("Finally Friday");
    break;
case 6:
    document.write("Super Saturday");
    break;
case 0:
    document.write("sleepy Sunday");
    break;
default:
    document.write("I'm looking forward to this weekend!");
}
</script>
</body>
</html>

```

**Output**

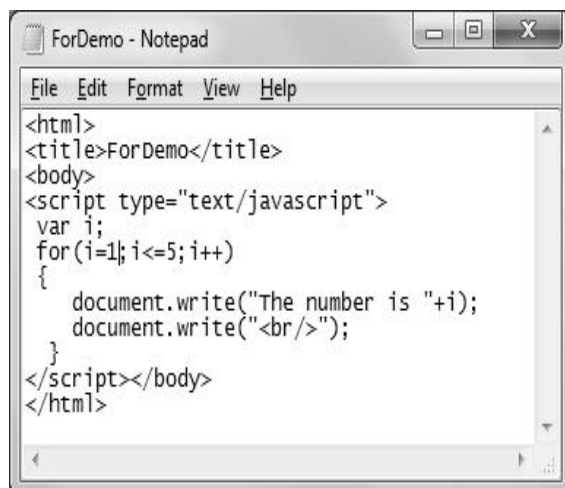
Loop statements

- Loops execute a block of code a specified number of times, or while a specified condition is true.
- Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.
- In JavaScript, there are two different kind of loops:
  - for - loops through a block of code a specified number of times. It can be used only when it is known in advance, how many times we have to run the loop.
  - while - loops through a block of code while a specified condition is true.
- For Loop Syntax:

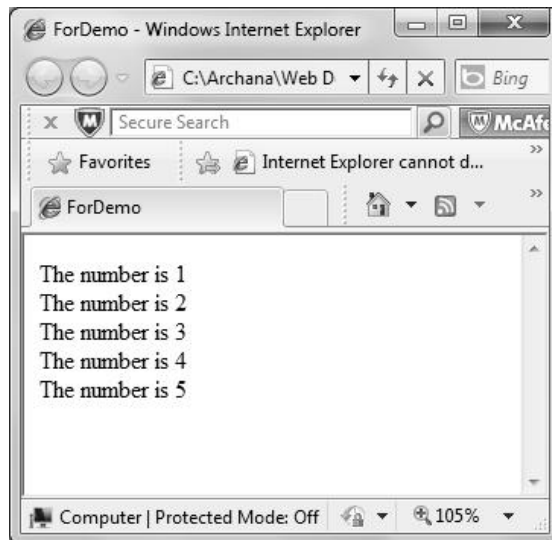
```
for  
(variable=startvalue;variable<=endvalue;variabl  
e=variable+increment)  
{  
code to be executed  
}
```

- **Example:**

Javascript given below will print 5 numbers. Each time, value of the variable is incremented by 1.



```
File Edit Format View Help  
<html>  
<title>ForDemo</title>  
<body>  
<script type="text/javascript">  
var i;  
for(i=1;i<=5;i++)  
{  
    document.write("The number is "+i);  
    document.write("<br/>");  
}  
</script></body>  
</html>
```



- While loop syntax
 

```
while (var<=endvalue)
{
    code to be executed
}
```
- While loop can be used with any comparison operator.
- **Do While** loop is a variation to the while loop. In this case block will be executed at least once, as the statements are executed before the condition is tested. Syntax for do while is as follows –

```
do
{
    code to be executed
}
while (var<=endvalue);
```

- Consider the example where number is printed after incrementing it by 1. This is performed while the number is less than or equal to 5. Script and the outputs with while and do while loop are as given below –

```
<html>
<body>
<script type="text/javascript">
var i=6;
do
{
    document.write("The number is " + i);
    document.write("<br />");
    i++;
}
while (i<=5);
```

```

</script>
</body>
</html>

```

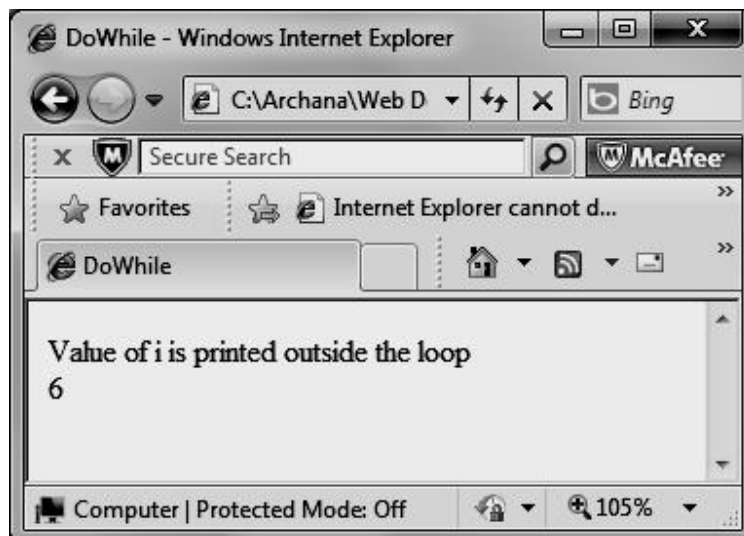


- Changes in script with just while loop and corresponding output --
 

```

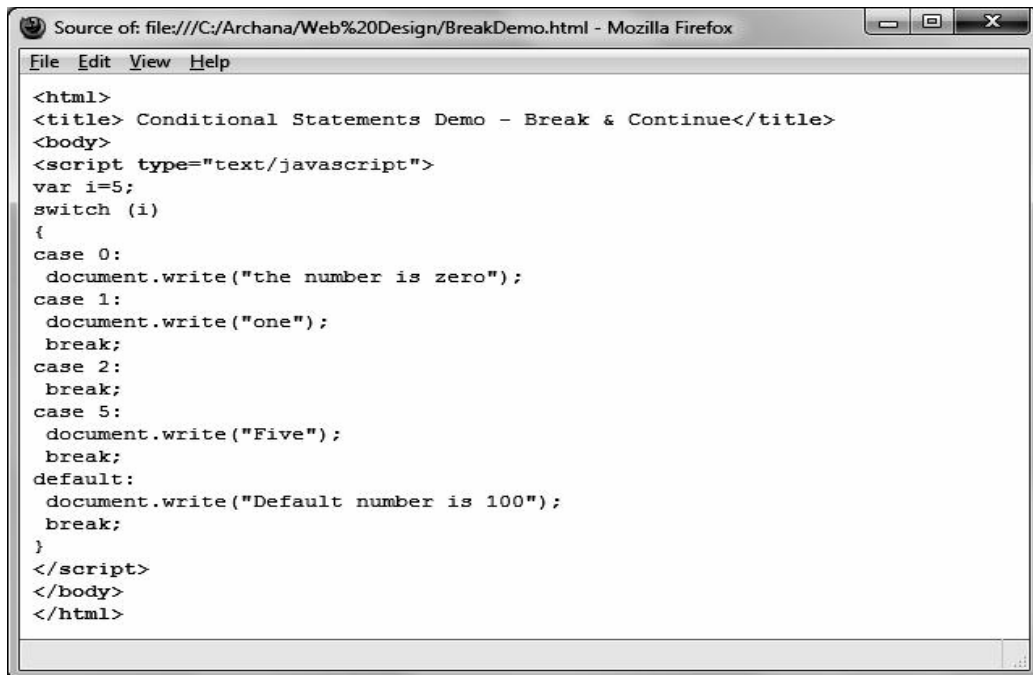
while (i<=5)
{
    document.write("The number is " + i);
    document.write("<br />");
    i++;
}
document.write("value of i is printed
outside the loop");
document.write("<br>\\" + i);

```

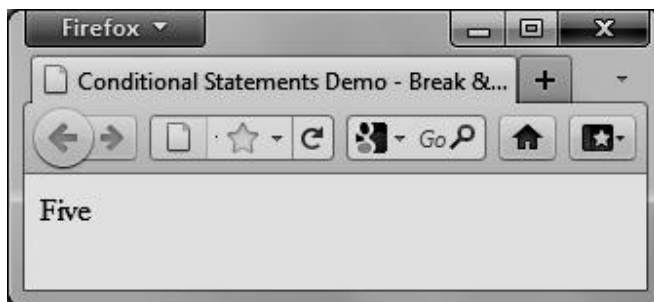


### Break and Continue statement in javascript

- The break statement will break the loop and continue executing the code that follows after the loop (if any).
- The continue statement will break the current loop and continue with the next value.



```
Source of: file:///C:/Archana/Web%20Design/BreakDemo.html - Mozilla Firefox
File Edit View Help
<html>
<title> Conditional Statements Demo - Break & Continue</title>
<body>
<script type="text/javascript">
var i=5;
switch (i)
{
case 0:
document.write("the number is zero");
case 1:
document.write("one");
break;
case 2:
break;
case 5:
document.write("Five");
break;
default:
document.write("Default number is 100");
break;
}
</script>
</body>
</html>
```



### Function

- Function is a segment of program that performs a given task.
- A function contains code that will be executed by an event or by a call to the function.
- You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external javascript file).
- Functions can be defined both in the <head> and in the <body> section of a document.



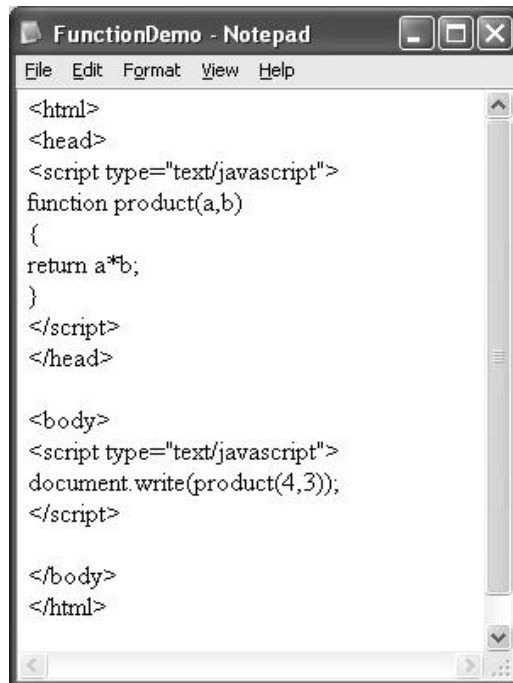
- However, to assure that a function is read/loaded by the browser before it is called, it should be put functions in the <head> section.
- Syntax of the function is as follows –

```
function functionname(var1,var2,...,varX)
{
  some code
}
```

- Function always includes parenthesis after the name of function ' ( ) '
- Function calls are case sensitive as javascript is case sensitive.
- The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

- If a variable is declared using "var", within a function, the variable can only be accessed within that function.
- The variable is destroyed once function call is over. These variables are called local variables. Local variables can have the same name in different functions, because each is recognized only by the function in which it is declared.
- If a variable is declared outside a function, all the functions on your page can access it.
- The lifetime of these variables starts when they are declared, and ends when the page is closed.
- Following is an example of function. Function products computes and returns product of variables a and b.
- Basic advantage of using a function is reusability. Same task can be performed again and again simply by calling the function which performs the task.

**Example:**

```
FunctionDemo - Notepad
File Edit Format View Help
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>

<body>
<script type="text/javascript">
document.write(product(4,3));
</script>

</body>
</html>
```



---

## 7.5 CORE JAVASCRIPT

---

Data Types are classified as primitive data types and composite data types. Composite Data types -

- Numbers - are values that can be processed and calculated. The numbers can be either positive or negative. Javascript integer can have three base values – 10 (decimal), 8(octal) or 16(hexadecimal). Number can be integer or floating point number.

- Strings - are a series of letters and numbers enclosed in single or double quotation marks. Strings are used for text to be displayed or values to be passed along.
- Some characters that you may want in a string may not exist on the keyboard, or may be special characters that can't appear as themselves in a string.
- In order to put these characters in a string, you need to use an escape sequence to represent the character. An escape sequence is a character or numeric value representing a character that is preceded by a backslash ( \ ) to indicate that it is a special character.

Some escaped characters are as follows:

Escape Sequence	Character
\b	Backspace.
\t	Tab. Tabs behave erratically on the Web and are best avoided, but sometimes you need them.
\n	New line (\u000a). Inserts a line break at the point specified. It is a combination of the carriage return (\r) and the form feed (\f).
\"	Double quote.
\'	Single quote, or an apostrophe, such as in <code>can\'t</code> .
\\	The backslash, since by itself it is a command.

- Boolean (true/false) - lets you evaluate whether a condition meets or does not meet specified criteria.
- Null - is an empty value. null is not the same as 0 -- 0 is a real, calculable number, whereas null is the absence of any value. An empty string is distinct from null value.
- NAN – Some javascript functions return a special value called not a number.

Primitive Data Types –  
Object –

- An *object* is a collection of named values, called the *properties* of that object. Functions associated with an object are referred to as the *methods* of that object.

- Properties and methods of objects are referred to with a dot(.) notation that starts with the name of the object and ends with the name of the property.  
For instance  
image.src.
- Normally in objects there are only two nodes, the object and the property, but sometimes the properties can have properties of their own, creating an object tree.
- For instance, document.form1.namefield.
- Objects in JavaScript can be treated as associative arrays. This means that image.src and image['src'] are equivalent.
- JavaScript has many predefined objects, such as a Date object and a Math object. These are used much as function libraries are used in a language like C.
- They contain a collection of useful methods that are predefined and ready for use in any JavaScript code you may write.

#### Date Object –

- The Date object is used to work with dates and times.
- Date objects are created with new Date().
- There are four ways of instantiating a date:
 

```
var d = new Date();
var d = new Date(milliseconds);
var d = new Date(dateString);
var d = new Date(year, month, day, hours,
minutes,
seconds, milliseconds);
```
- Some javascript predefined methods are –
  - **getDate()** Returns the day of the month (from 1-31)
  - **getDay()** Returns the day of the week (from 0-6)
  - **getFullYear()** Returns the year (four digits)
  - **getHours()** Returns the hour (from 0-23)
  - **getMilliseconds()** Returns the milliseconds (from 0-999)
  - **getMinutes()** Returns the minutes (from 0-59)
  - **getMonth()** Returns the month (from 0-11)
  - **getSeconds()** Returns the seconds (from 0-59)
  - **getTime()** Returns the number of milliseconds since midnight Jan 1, 1970

- **getTimezoneOffset()** Returns the time difference between GMT and local time, in minutes
- **getYear()** Deprecated. Use the
- **getFullYear()** method instead
- **parse()** Parses a date string and returns the number of milliseconds since midnight of January 1, 1970
- **setDate()** Sets the day of the month (from 1-31)
- **setHours()** Sets the hour (from 0-23)
- **setMilliseconds()** Sets the milliseconds (from 0-999)
- **setMinutes()** Set the minutes (from 0-59)
- **setMonth()** Sets the month (from 0-11)
- **setSeconds()** Sets the seconds (from 0-59)
- **setTime()** Sets a date and time by adding or subtracting a specified number of milliseconds to/from midnight January 1, 1970
- **toString()** Converts a Date object to a string
- **toTimeString()** Converts the time portion of a Date object to a string
- **toUTCString()** Converts a Date object to a string, according to universal time
- **valueOf()** Returns the primitive value of a Date object

#### **Math object –**

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical constants and methods.
  - **round()** – rounds the number to nearest integer value.
  - **random()** - returns a random number between 0 and 1.
  - **max()** - returns the number with the highest value of two specified numbers.
  - **min()** - returns the number with the lowest value of two specified numbers.

**Array –**

- An *Array* is an ordered collection of data values.
- In JavaScript, an array is just an object that has an index to refer to its contents. In other words, the fields in the array are numbered, and you can refer to the number position of the field.
- The array index is included in square brackets immediately after the array name. In JavaScript, the array index starts with zero, so the first element in an array would be `arrayName[0]`, and the third would be `arrayName[2]`.
- JavaScript does not have multi-dimensional arrays, but you can nest them, which is to say, an array element can contain another array.
- You access them listing the array numbers starting for the outmost array and working inward. Therefore, the third element (position 2) of or inside the ninth element (position 8) would be `arrayName[8][2]`.

**7.6 Summary**

- JavaScript is light weighted programming language used for small applications.
- Javascript is object based programming language made in C++.
- Javascript can be executed either on client side browser or on server side browser.
- Javascripts can be downloaded and run and carry a threat of virus attack. Security of javascript depends upon the security of java language. The process of executing the javascript code after isolating it from the operating system is called 'sandbox' model.
- Different types of javascript operators help perform various arithmetic, logical, string and other types of functions.
- Browser sequentially executes javascript statements of various types such as conditional and loop
- statements. Some of loop statements are while, dowhile, switch statement etc.
- Number, Boolean, string are few data types used in javascript.

---

**7.7 EXERCISE**

---

1. Define javascript. How is it used?
2. explain the difference between client side and server side javascript.
3. explain how virus threat from javascript is avoided?
4. write a javascript to find greatest of three given numbers
5. write a javascript to print the month if numeric value for the month is given (eg January for 1 and so on)
6. write a javascript to print all prime numbers from 1 to 100 in reverse order.
7. write a javascript to print reverse of a number.
8. explain what is an object in javascript. Describe the date and math object with at least two properties each.
9. explain the difference between String"" and null value
10. discuss what is a function? Does it always return a value? Give example of at least one function.
11. what is an array? How is it used in javascript?
12. explain how a date value can be converted into a string value and vice versa.



## JAVASCRIPT – 2

### Unit Structure

#### 8.0 Objective

#### 8.1 Document and associated objects

8.1.1 Document

8.1.2 Link

8.1.3 Area

8.1.4 Anchor

8.1.5 Image

8.1.6 Applet

8.1.7 Layer

#### 8.2 Events and Event Handlers

8.2.1 General information about events

8.2.2 Defining event handlers

#### 8.3 Event

8.3.1 onAbort

8.3.2 onBlur

8.3.3 onChange

8.3.4 onClick

8.3.5 onDblClick

8.3.6 onDragDrop

8.3.7 onError

8.3.8 onFocus

8.3.9 onKeyDown

8.3.10 onKeyPress

8.3.11 onKeyUp

8.3.12 onLoad

8.3.13 onMouseDown

8.3.14 onMouseMove

8.3.15 onMouseOut



- 8.3.16 onmouseover
- 8.3.17 onMouseUp
- 8.3.18 onMove
- 8.3.19 onReset
- 8.3.20 onResize
- 8.3.21 onSelect
- 8.3.22 onSubmit
- 8.3.23 onUnload

---

## **8.1 OBJECTIVE**

---

After reading this chapter you will be able to -

- Understand the use of document object and other objects associated with the document object.
- Understand and use the properties and methods and to use common properties and methods.
- Understand and use objects such as area, anchor, link etc. and properties and methods associated with these objects.
- Write javascripts using the above properties and methods.
- Understand the concept of event and event handlers.
- Understand the use of each event associated with the document object.

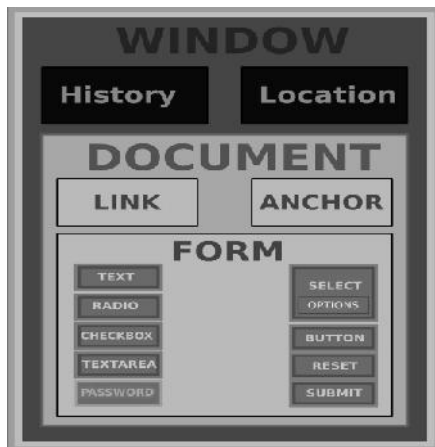
---

## **8.1 DOCUMENT AND ASSOCIATED OBJECTS**

---

### **8.1.1 Document**

- Each HTML document loaded into a browser window becomes a Document object.
- The Document object provides access to all HTML elements in a page, from within a script.
- The Document object is also part of the Window object, and can be accessed through the 'window.document' property.
- Document object is part of document object model.
- This model has a fixed hierarchy, where topmost object in the hierarchy is Browser itself.
- After browser window object and inside window, as shown below, comes the document object.
- Relation of document object to the window object can be depicted in the fig given below –



```

-> Document
    |-> Anchor
    |-> Link
    |-> Images
    |-> Tags
    |-> Form

-> Text-box
-> Text Area
-> Radio Button
-> Check Box
-> Select
-> Button

```

Fig 8.1 Document Object and Window object      Fig 8.2 Hierarchy in Document Object

- Document object has following properties and methods

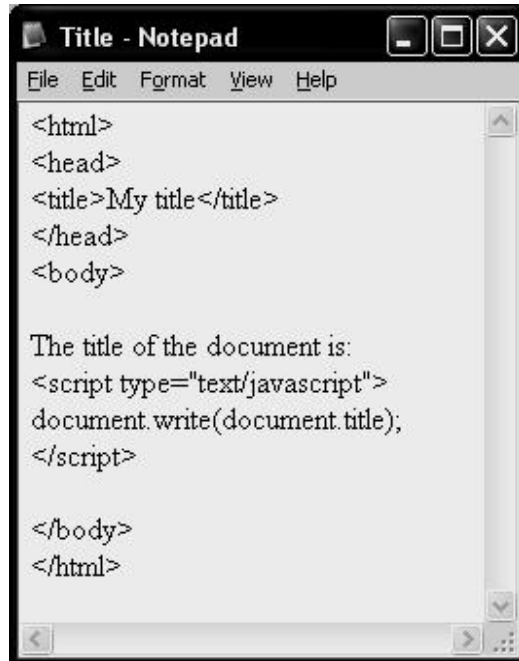
#### Properties –

- **cookie** Returns all name/value pairs of cookies in the document
- **documentMode** Returns the mode used by the browser to render the document
- **domain** Returns the domain name of the server that loaded the document
- **lastModified** Returns the date and time the document was last modified
- **readyState** Returns the (loading) status of the document
- **referrer** Returns the URL of the document that loaded the current document
- **title** Sets or returns the title of the document
- **URL** Returns the full URL of the document

#### Methods –

- **close()** Closes the output stream previously opened with document.open()
- **getElementById()** Accesses the first element with the specified id
- **getElementsByName()** Accesses all elements with a specified name
- **getElementsByTagName()** Accesses all elements with a specified tagname
- **open()** Opens an output stream to collect the output from document.write() or document.writeln()
- **write()** Writes HTML expressions or JavaScript code to a document
- **writeln()** Same as write(), but adds a newline character after each statement Yes

Following examples demonstrates use of some properties and methods of document object.



```
<html>
<head>
<title>My title</title>
</head>
<body>

The title of the document is:
<script type="text/javascript">
document.write(document.title);
</script>

</body>
</html>
```



**Fig 8.3 Script and output Demonstrating write method and title property of Document object.**



```

OpenMethod - Notepad
File Edit Format View Help
<html>
<head>
<script type="text/javascript">
function createDoc()
{
var doc=document.open("text/html","replace");
var txt="<html><body>Example of creating a document object for writing text. </body></html>";
doc.write(txt);
doc.close();
}
</script>
</head>

<body>
<input type="button" value="New document" onclick="createDoc()" />
</body>
</html>

```

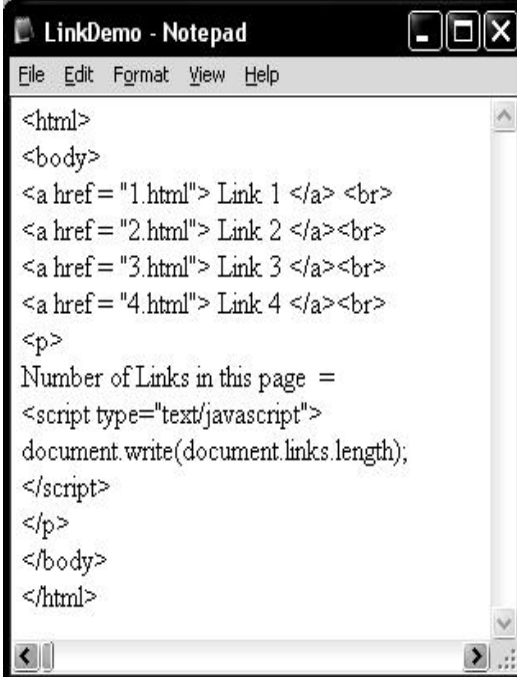
**Fig 8.4 Script Demonstrating the Document.open() method to open a textstream**



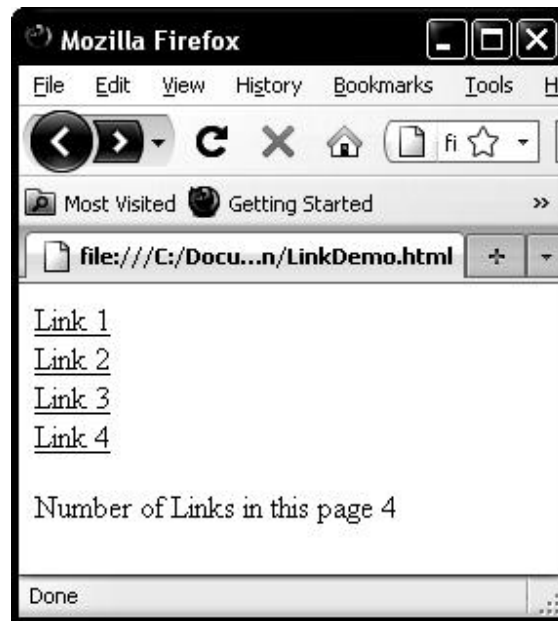
**Fig 8.5 Output of the script Before and After Clicking the 'New Document' button.**

### 8.1.2 Link

- The Link object represents an HTML link element.
- The link element must be placed inside the head section of an HTML document, and it specifies a link to an external resource.
- In other words, link tag defines how a click on object can redirect or take the browser window to a new location specified by the address specified.
- A common use of the <link> tag is to link to external style sheets.
- Some of the properties of javascript link objects are as given below—
  - charset**: - Sets or returns the character encoding of a linked document
  - href**: - Sets or returns the URL of a linked document
  - hreflang** :- Sets or returns the language code of the linked document
  - media**: - Sets or returns the media type for the link element
  - type** : - Sets or returns the content type of the linked document
- In addition to above properties, link object supports all standard properties like id, length, chartype etc.



```
<html>
<body>
<a href = "1.html"> Link 1 </a> <br>
<a href = "2.html"> Link 2 </a><br>
<a href = "3.html"> Link 3 </a><br>
<a href = "4.html"> Link 4 </a><br>
<p>
Number of Links in this page =
<script type="text/javascript">
document.write(document.links.length);
</script>
</p>
</body>
</html>
```



**Fig 8.6 Script and Outputs Demonstrating length property of link**

### 8.1.3 Area

- The Area object represents an area inside an HTML image-map (an image-map is an image with clickable areas).
- For each <area> tag in an HTML document, an Area object is created.
- In addition to standard properties and methods, javascript area object supports following properties –

**alt** Sets or returns the value of the alt attribute of an area

**coords** Sets or returns the value of the coords attribute of an area

**hash** Sets or returns the anchor part of the href attribute value

**host** Sets or returns the hostname:port part of the href attribute value

**hostname** Sets or returns the hostname part of the href attribute value

**href** Sets or returns the value of the href attribute of an area

**noHref** Sets or returns the value of the nohref attribute of an area

**pathname** Sets or returns the pathname part of the href attribute value

**port** Sets or returns the port part of the href attribute value

**protocol** Sets or returns the protocol part of the href attribute value

**search** Sets or returns the querystring part of the href attribute value

**shape** Sets or returns the value of the shape attribute of an area

**target** Sets or returns the value of the target attribute of an area

- Following example shows an image map and returns the shape of an area marked in the image map.

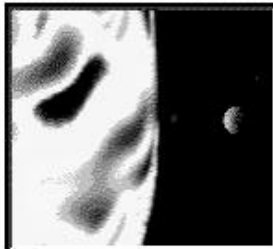
```

PlanetMap - Notepad
File Edit Format View Help
<html>
<body>


<map name="planetmap">
<area id="venus" shape="circle" coords="124,58,8"
alt="Venus"
href="venus.htm" />
</map>
<p>The shape for the "Venus" area is:
<script type="text/javascript">
document.write(document.getElementById("venus").shape);
</script>
</p>

</body>
</html>

```



The shape for the "Venus" area is: circle

**Fig 8.7 Script and output Demonstrating Area object.**

#### 8.1.4 Anchor

- The Anchor object represents an HTML hyperlink.
- For each <a> tag in an HTML document, an Anchor object is created.
- An anchor allows you to create a link to another document (with the href attribute), or to a different point in the same document (with the name attribute).

- You can access an anchor by using `getElementById()`, or by searching through the `anchors[]` array of the Document object.
- In addition to standard properties and methods, javascript anchor object supports following properties –

**charset** Sets or returns the value of the charset attribute of a link

**href** Sets or returns the value of the href attribute of a link

**hreflang** Sets or returns the value of the hreflang attribute of a link

**name** Sets or returns the value of the name attribute of a link

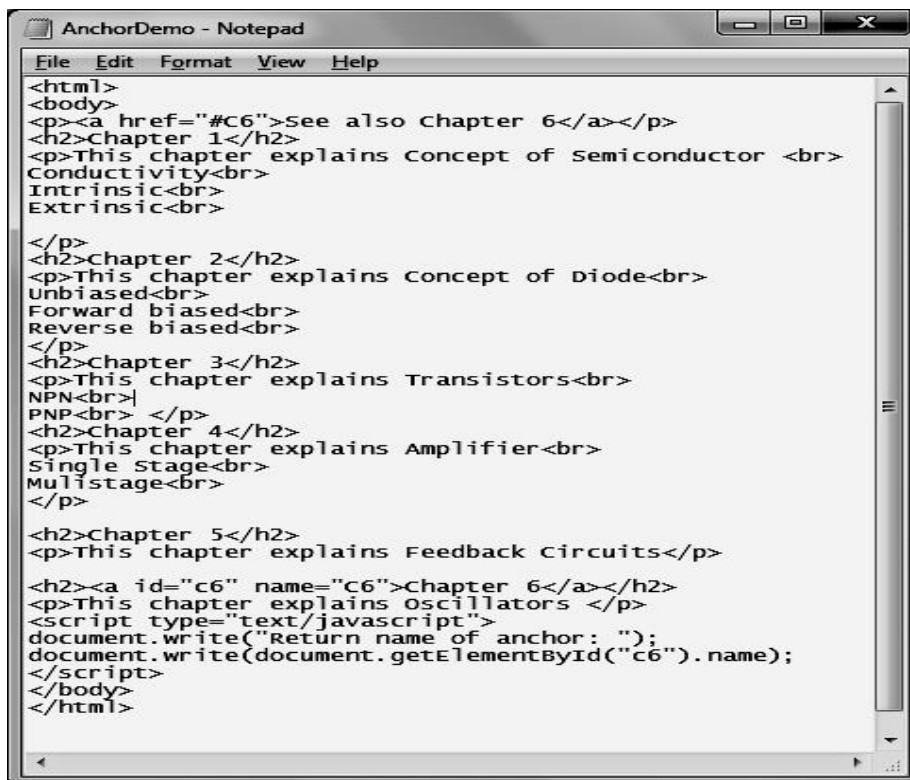
**rel** Sets or returns the value of the rel attribute of a link

**rev** Sets or returns the value of the rev attribute of a link

**target** Sets or returns the value of the target attribute of a link

**type** Sets or returns the value of the type attribute of a link

- Following example shows anchor tag used in a web page.



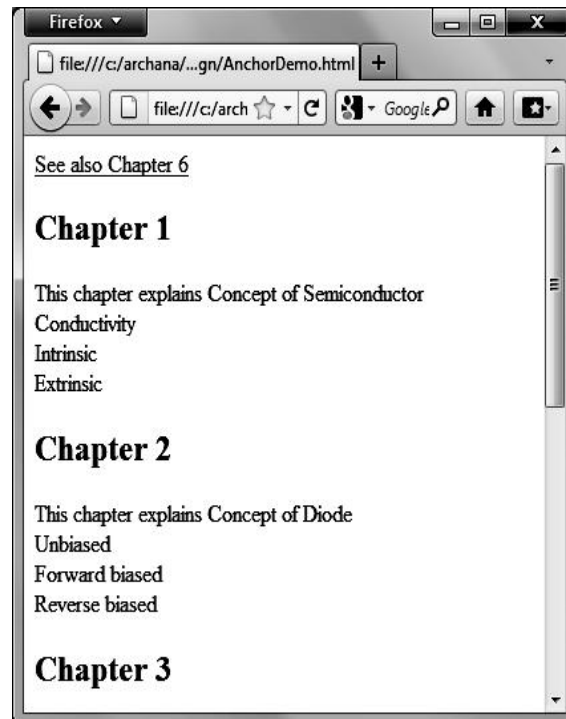
```

AnchorDemo - Notepad
File Edit Format View Help
<html>
<body>
<p><a href="#C6">See also Chapter 6</a></p>
<h2>Chapter 1</h2>
<p>This chapter explains Concept of Semiconductor <br>
Conductivity<br>
Intrinsic<br>
Extrinsic<br>
</p>
<h2>Chapter 2</h2>
<p>This chapter explains Concept of Diode<br>
Unbiased<br>
Forward biased<br>
Reverse biased<br>
</p>
<h2>Chapter 3</h2>
<p>This chapter explains Transistors<br>
NPN<br>
PNP<br>
</p>
<h2>Chapter 4</h2>
<p>This chapter explains Amplifier<br>
Single stage<br>
Multistage<br>
</p>
<h2>Chapter 5</h2>
<p>This chapter explains Feedback Circuits</p>
<h2><a id="c6" name="C6">Chapter 6</a></h2>
<p>This chapter explains Oscillators </p>
<script type="text/javascript">
document.write("Return name of anchor: ");
document.write(document.getElementById("c6").name);
</script>
</body>
</html>

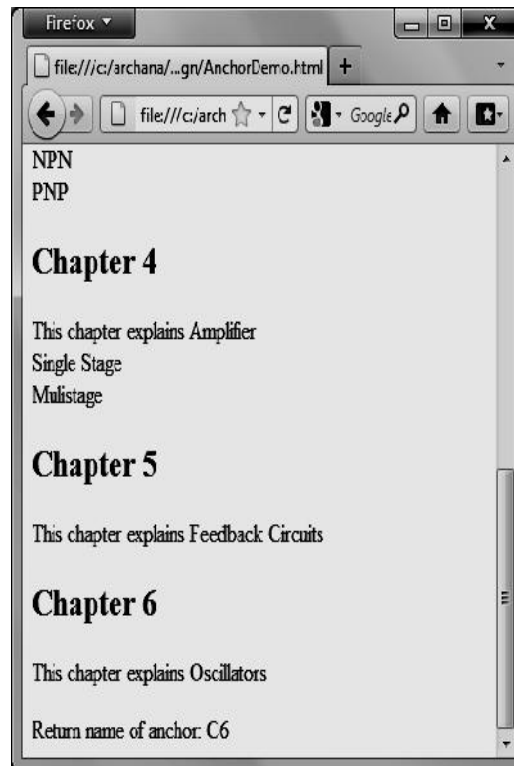
```

Fig 8.8 Script For demonstrating anchor object





<b><u>align</u></b>	Sets or returns the value of the align attribute of an image
<b><u>alt</u></b>	Sets or returns the value of the alt attribute of an image
<b><u>border</u></b>	Sets or returns the value of the border attribute of an image
<b><u>complete</u></b>	Returns whether or not the browser is finished loading an image
<b><u>height</u></b>	Sets or returns the value of the height attribute of an image
<b><u>hspace</u></b>	Sets or returns the value of the hspace attribute of an image
<b><u>longDesc</u></b>	Sets or returns the value of the longdesc attribute of an image
<b><u>lowsrc</u></b>	Sets or returns a URL to a low-resolution version of an image
<b><u>name</u></b>	Sets or returns the name of an image
<b><u>src</u></b>	Sets or returns the value of the src attribute of an image
<b><u>useMap</u></b>	Sets or returns the value of the usemap attribute of an image
<b><u>vspace</u></b>	Sets or returns the value of the vspace attribute of an image
<b><u>width</u></b>	Sets or returns the value of the width attribute of an image



**Fig 8.9 Before and after using the anchor link.**

- After the anchor is used URL of browser changed to 'file:///c:/AnchorDemo.html#C6', as c6 is used as an anchor.

### 8.1.5 Image

- The Image object represents an embedded image.
- For each <img> tag in an HTML document, an Image object is created.
- Notice that images are not technically inserted into an HTML page, images are linked to HTML pages. The <img> tag creates a holding space for the referenced image.
- Image object, in addition to the standard properties and methods, supports following properties and events.
- Image Object Events

onabort Loading of an image is interrupted

onerror An error occurs when loading an image

onload An image is finished loading

- Following is the example of some properties of image object and javascript functions used to change these properties.
- Function changesize changes the height and width of the image object and function addBorder adds border to the image.
- Similarly, we can also change the image by using the src attribute of image tag.
- Alt attribute gives the text to be displayed, if the image can not be displayed.
- Also, a lower version of image to speed up loading, can be shown using a lowsrc option attribute.

### Example:

```

imageDemo - Notepad
File Edit Format View Help
<html>
<head>
<script type="text/javascript">
function addBorder()
{
document.getElementById("compgirl").border="2";
}
function changeSize()
{
document.getElementById("compgirl").height="250";
document.getElementById("compgirl").width="300";
}
</script>
</head>
<body>

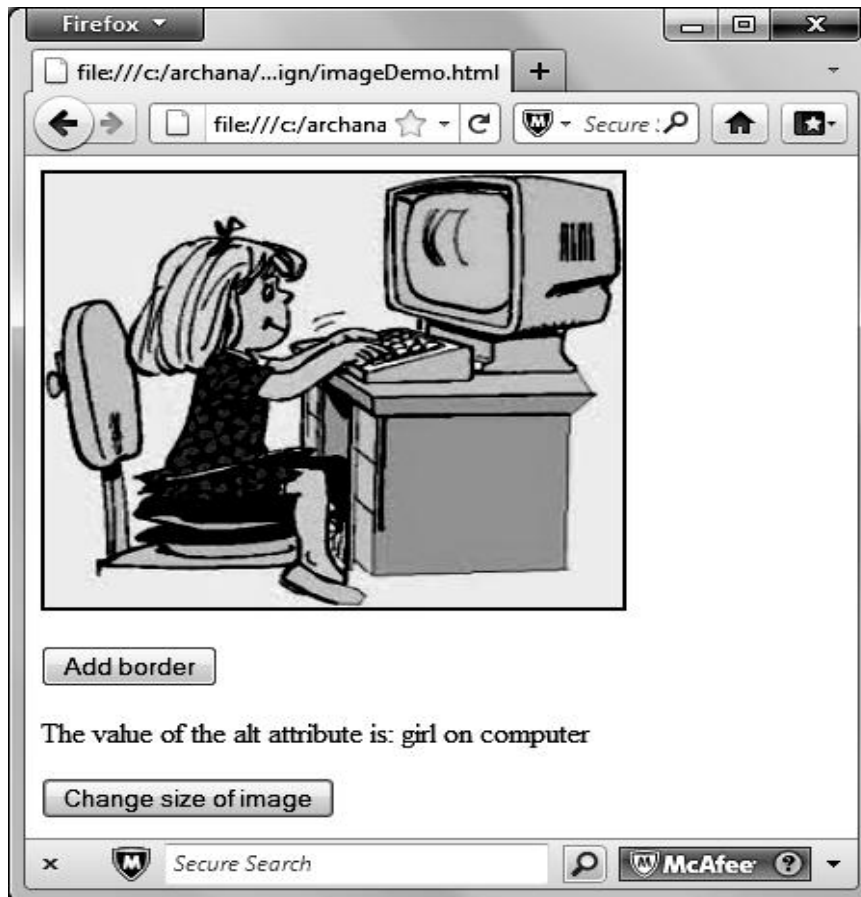
<br /><br />
<input type="button" onclick="addBorder()" value="Add border" />
<p>
The value of the alt attribute is:
<script type="text/javascript">
document.write(document.getElementById("compgirl").alt);
</script>
</p>
<input type="button" onclick="changeSize()" value="Change size of image" />
</body>
</html>

```

Fig 8.10 source code for demonstrating some image properties and attributes.



Fig 8.11 Before click of the command button objects



**Fig 8.12** After click of the command buttons, border added and a increased sized image.

### 8.1.6 Applet

- Applet tag was used in earlier version of HTML (HTML4) to embed an java applet in a browser. It is not supported in HTML 5.0 and is replaced by the object tag.
- The <object> tag defines an embedded object within an HTML document. It is used to embed multimedia (like audio, video, Java applets, ActiveX, PDF, and Flash) in your web pages.
- You can also use the <object> tag to embed another webpage into your HTML document.
- You can use the <param> tag to pass parameters to plugins that have been embedded using the <object> tag.
- An object element must appear inside the body element. The text between the <object> and </object> is an alternate text, for browsers that do not support this tag.

- At least one of the "data" and "type" attributes MUST be defined where data specifies a URL that refers to the object's data and type Specifies the MIME type of data specified in the data attribute. (Multipurpose Internet Mail Extensions (**MIME**) is an Internet standard that extends the format of email to support Text in character sets other than ASCII, Non-text attachments, Message bodies with multiple parts, Header information in non-ASCII character sets.

---

## 8.2 EVENTS AND EVENT HANDLERS

---

### 8.2.1 General information about events

- Events are actions that can be detected by JavaScript.
- Every element on a web page has certain events which can trigger a JavaScript.
- For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button.
- Examples of events:
  - A mouse click
  - A web page or an image loading
  - Mousing over a hot spot on the web page
  - Selecting an input field in an HTML form
  - Submitting an HTML form
  - A keystroke
- Events are normally used in combination with functions, and the function will not be executed before the event occurs!

### 8.2.2 Defining event handlers

- They are JavaScript code that are not added inside the <script> tags, but rather, inside the html tags, that execute JavaScript when something happens, such as pressing a button, moving your mouse over a link, submitting a form etc.
- The basic syntax of these event handlers is:

```
name_of_handler="JavaScript code here"
```

- For example:

```
<a href="http://google.com"
onClick="alert( 'hello!' )" >Google</a>
```

- When events are associated with functions, the functions are written in the head section within the <script> tag and are called from the event handlers.

**Example :**

```

<html>
<body>
<h1      onclick="this.innerHTML='Welcome      to
EventHandlers'">Click      on      this      text</h1>
</body>
</html>

```

- This code will generate following output –



**Fig 8.12 Event Handler and after event is called.**

---

### 8.3 EVENT

---

Following is the list of events used by various javascript objects and when are these events triggered.

Attribute	The event occurs when...
onabort	Page is not finished loading
<a href="#">onblur</a>	An element loses focus
<a href="#">onchange</a>	The content of a field changes
<a href="#">onclick</a>	Mouse clicks an object
<a href="#">ondblclick</a>	Mouse double-clicks an object
ondragdrop	A user drops an object
<a href="#">onerror</a>	An error occurs when loading a document or an image

<a href="#">onfocus</a>	An element gets focus
<a href="#">onkeydown</a>	A keyboard key is pressed
<a href="#">onkeypress</a>	A keyboard key is pressed or held down
<a href="#">onkeyup</a>	A keyboard key is released
<a href="#">onload</a>	A page or image is finished loading
<a href="#">onmousedown</a>	A mouse button is pressed
<a href="#">onmousemove</a>	The mouse is moved
<a href="#">onmouseout</a>	The mouse is moved off an element
<a href="#">onmouseover</a>	The mouse is moved over an element
<a href="#">onmouseup</a>	A mouse button is released
onmove	The position of top left corner of an object is moved.
<a href="#">onresize</a>	A window or frame is resized
onreset	Reset button on the form is clicked.
<a href="#">onselect</a>	Text is selected
onsubmit	Validate all form fields before submitting it.
<a href="#">onunload</a>	The user exits the page

- Following is an example which shows onmouseover event to give different messages for different parts of an imagemap.

```

eventDemo Notepad
File Edit Format View Help
<html>
<head>
<script type="text/javascript">
function writeText(txt)
{
document.getElementById("desc").innerHTML+=txt;
}
</script>
</head>
<body>

<map name="planetmap">
<area shape="rect" coords="0,0,82,126"
onmouseover="writeText('The Sun and the gas giant planets like Jupiter are by far the largest objects in
our Solar System.')"
href="sun.htm" target="_blank" alt="Sun" />
<area shape="circle" coords="90,58,3"
onmouseover="writeText('The planet Mercury is very difficult to study from the Earth because it is
always so close to the Sun.')"
href="mercur.htm" target="_blank" alt="Mercury" />
<area shape="circle" coords="124,58,8"
onmouseover="writeText('Until the 1960s, Venus was often considered a twin sister to the Earth
because Venus is the nearest planet to us, and because the two planets seem to share many
characteristics.')"
href="venus.htm" target="_blank" alt="Venus" />
</map>
<p id="desc">Mouse over the sun and the planets and see the different descriptions.</p>
</body>
</html>

```

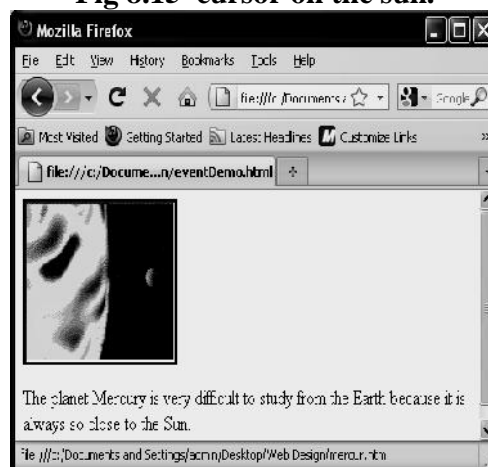
**Fig 8.13 Script to demonstrate onmouseover event and its event handler**



**Fig 8.14 Newly Loaded page**

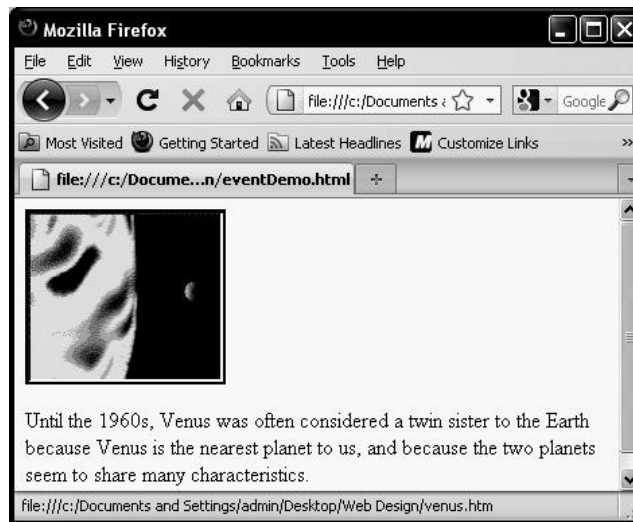


**Fig 8.15 cursor on the sun.**



**Fig 8.16 Cursor pointing to mercury**





**Fig 8.17 Cursor pointing venus.**

---

## 8.4 EXERCISES

---

1. Define document object. What are the objects associated with document.
2. which are the different methods associated with document object.
3. write a javascript to write a message "Welcome to the Document object" using the write method of document object.
4. list the objects associated with form object.
5. Explain what is link object? What is the use of link object? State any two properties of this object.
6. List various properties of Area tag. What is the use of shape property of area object? Explain how is it used with an example.
7. What is anchor? How is it used in a document object?
8. Define an event. What is event handler? Explain how event handler is used.
9. Explain how a function can be used as event handler?
10. Write a javascript with at least three functions to demonstrate use of onmousemove and onmouseover event.
11. Write a javascript to display a message "welcome to scripting" when a button labled "hello" is clicked. Display a message "Thank you for using scripting" when the same button is doubleclicked.
12. Write a javascript code to display a text "Image can not be displayed in current browser setting" if a image used in script can not be disaplyed.



# XML

## Unit Structure

- 9.0. Objective
- 9.1. Introduction to XML
- 9.2. Anatomy of an XML Document
- 9.3. Creating XML Documents
- 9.4. XML DTDs
- 9.5. XML Schemas
- 9.6. XSL
- 9.7. Exercise

---

## 9.0. OBJECTIVE

---

After going through this chapter you will be able to:

- Explain the purpose of XML
- Identify benefits of using XML
- Explain the format of XML document
- Identify different built-in data types
- Identify XML syntax rules
- Explain role of DTDs and how to create DTDs
- Explain what is XML Schema
- Identify the importance of XSL

---

## 9.1. INTRODUCTION TO XML

---

XML (extensible Markup Language) is a meta-language; that is, it is a language in which other languages are created. In XML, data is "marked up" with tags, similar to HTML tags. In fact, the latest version of HTML, called XHTML, is an XML-based language, which means that XHTML follows the syntax rules of XML.

XML was designed to describe data. XML is used to store data or information. This data might be intended to be by read by people or by machines. It can be highly structured data such as data typically stored in

databases or spreadsheets, or loosely structured data, such as data stored in letters or manuals. XML tags are not predefined in XML. You must define your own tags. XML uses a DTD (Document Type Definition) to formally describe the data.

The main difference between XML and HTML

1. XML is not a replacement for HTML.  
XML and HTML were designed with different goals:
  - a. XML was designed to describe data and to focus on what data is.
  - b. HTML was designed to display data and to focus on how data looks.
2. HTML is about displaying information, XML is about describing information.

### **XML Benefits**

When you write an HTML document, you see a nicely formatted page in a browser - instant satisfaction. When you write an XML document, you see an XML document (not the output) - not so exciting.

1. **XML Holds Data, Nothing More**

XML does not really do much of anything. Rather, developers can create XML-based languages that store data in a structure way. Applications can then use this data to do any number of things.

2. **XML Separates Structure from Formatting**

One of the difficulties with HTML documents, word processor documents, spreadsheets, and other forms of documents is that they mix structure with formatting. This makes it difficult to manage content and design, because the two are mix together.

In HTML, there is a <U> tag used for underlining text. It is also used for emphasis, or to mark a unit title. It would be very difficult to write an application that searches through such a document for unit titles.

In XML, the book titles could be placed in <UNIT\_TITLE> tags and the emphasized text could be place in <EM> tags.

3. **XML Promotes Data Sharing**

Applications that hold data using different structures must share data with one another. It can be very difficult for a developer to map the different data structures to each other. XML can solve this problem. Each application's data structure is mapped to an agreed-upon XML structure. Then all the applications share data in this XML format. Each application only has to know two structures, its own and the XML structure, to be able to share data with many other applications.

#### 4. XML is Human-Readable

XML documents are (or can be) read by people as data stored in a database. It is not easy to browse through a database and read different segments of it as you would a text file. Given below is an XML document (person.xml):

```
<?XML version="1.0"?>
<PERSON>
<NAME>
<FIRSTNAME>Raj</FIRSTNAME>
<LASTNAME>Mehra</LASTNAME>
</NAME>
<JOB>Software Engineer</JOB>
<GENDER>Male</GENDER>
</PERSON>
```

##### Code Explanation

Above XML is describing a person named Raj Mehra, who is a software engineer and is male.

#### 5. XML is Free

XML doesn't cost anything to use. It can be written with a simple text editor or one of the many freely available XML authoring tools, such as XML Notepad. In addition, many web development tools, such as Dreamweaver and Visual Studio .NET have built-in XML support. There are also many free XML parsers, such as Microsoft's MSXML (downloadable from microsoft.com) and Xerces (downloadable at apache.org).

#### **Let us see the progress:**

1. \_\_\_\_\_ was designed to display data and to focus on how data looks.
  - a. HTML
  - b. SGML
  - c. XML
  - d. XHTML
2. XML separates \_\_\_\_\_ from formatting
  - a. data
  - b. document
  - c. structure
  - d. table

---

## **9.2. ANATOMY OF AN XML DOCUMENT**

---

An XML document is made up of the following parts:

#### 1. The Prolog (optional)

The prolog of an XML document can contain the following items.

- The XML Declaration

The XML declaration, if it appears at all, must appear on the very first line of the document with no preceding white space. It looks like this:

```
<?XML          VERSION="1.0"          ENCODING="UTF-8"
STANDALONE="yes"?>
```

This declares that the document is an XML document. The version attribute is required, but the encoding and standalone attributes are not. If the XML document uses any markup declarations that set defaults for attributes or declare entities then standalone must be set to no.

- Processing Instructions

Processing instructions are used to pass parameters to an application. These parameters tell the application how to process the XML document. For example, the following processing instruction tells the application that it should transform the XML document using the XSL stylesheet artist.xsl

```
<?XML-STYLESHREF="artist.xsl"
TYPE="text/xsl"?>
```

As shown above, processing instructions begin with and <? end with ?>.

- Comments

Comments can appear throughout an XML document. Like in HTML, they begin with <!-- and end with -->.

```
<!--This is a comment-->
```

- A Document Type Declaration

The Document Type Declaration (or DOCTYPE Declaration) has three roles.

1. It specifies the name of the document element.
2. It may point to an external Document Type Definition (DTD).
3. It may contain an internal DTD.

The DOCTYPE Declaration shown below simply states that the document element of the XML document is artists.

```
<!DOCTYPE ARTISTS>
```

If a DOCTYPE Declaration points to an external DTD, it must either specify that the DTD is on the same system as the XML document itself using SYSTEM keyword or that it is in some public location using PUBLIC keyword. It then points to the location of the DTD using a relative Uniform Resource Indicator (URI) or an absolute URI.

Syntax: <!--DTD is on the same system as the XML document-->

```
<!DOCTYPE ARTISTS SYSTEM "dtds/artists.dtd">
```

```
Syntax:  <!--DTD is publicly available-->
          <!DOCTYPE ARTISTS PUBLIC "-
          //freespace//DTD artists 1.0//EN"
          "http://www.freespace.com/artists/DTD/artists.dtd"
          >
```

In the second declaration above, public identifiers are divided into three parts:

1. An Organization (E.g., Freespace)
2. A Name for the DTD (E.g., Artists 1.0)
3. A Language (E.g., EN for English)

<b>Prolog (optional)</b>	
XML Declaration	<?XML VERSION="1.0" ENCODING="UTF-8" STANDALONE="no"?>
Document Type Definition (DTD)	<!DOCTYPE DOCUMENT SYSTEM "tts.dtd">
Comment	<!-- Here is a comment -->
Processing Instructions	<?XML-stylesheet TYPE="text/css" HREF="myStyles.css"?>
White Space	

## 2. The Document Element (usually containing nested elements)

- Document / Root Element

Every XML document must have at least one element, called the document/root element. The document element usually contains other elements, which contain other elements, and so on. Elements are denoted with tags. Consider again person.xml which we discussed earlier.

```
<?XML VERSION="1.0"?>
<PERSON>
  <NAME>
    <FIRSTNAME>Raj</FIRSTNAME>
    <LASTNAME>Mehra</LASTNAME>
  </NAME>
  <JOB>Singer</JOB>
  <GENDER>Male</GENDER>
</PERSON>
```

### **Code Explanation**

The document / root element is PERSON. It contains three elements: NAME, JOB and GENDER. Further, the NAME element contains two elements of its own: FIRSTNAME and LASTNAME. As you can see, XML elements are denoted with

tags, just as in HTML. Elements that are nested within another element are said to be children of that element.

- Empty Elements

In XML all elements might not contain other elements or text. E.g., in HTML, there is <IMG> element / tag used to display an image. It does not contain any text or elements / tags within it, so it is called an empty element. In XML, empty elements must be closed, but they do not require a separate close tag. Instead, they can be closed with a forward slash at the end of the open tag as shown below:

```
<IMG SRC="images/raj.jpg"/>
```

The above code is identical in function to the code below:

```
<IMG SRC="images/raj.jpg"></IMG>
```

Elements & Content (required)	
Root Element Opening Tag	<TTS>
Child Elements and Content	<pre>&lt;TT&gt;&lt;name&gt;XML &lt;/NAME&gt; &lt;URL&gt;http://www.myserver.com/xml/ tt&lt;/URL&gt;&lt;/TT&gt; &lt;TT&gt; &lt;NAME&gt;HTML &lt;/NAME&gt; &lt;URL&gt;http://www.myserver.com/html /tt&lt;/URL&gt;&lt;/TT&gt;</pre>
Root Element Closing Tag	</TTS>

- Attributes

XML elements can be further defined with attributes, which appear inside of the element's open tag as shown below:

```
<NAME TITLE="SoftwareEngineer">
```

```
<FIRSTNAME>Raj</FIRSTNAME>
```

```
<LASTNAME>Mehra</LASTNAME>
```

```
</NAME>
```

Here TITLE is an attribute of NAME element.

- CDATA

Sometimes it is necessary to include sections in an XML document that should not be parsed by the XML parser. These sections might contain content that will confuse the XML parser, perhaps because it contains content that appears to be XML, but is not meant to be interpreted as XML. Such content must be nested in CDATA sections. The syntax for CDATA sections is shown below:

```
<![CDATA[
```

This section will not get parsed by the XML parser.

```
]]>
```

- White Space

In XML data, there are only four white space characters. They are:

1. Tab
2. Line-feed
3. Carriage-return
4. Single space

There are several important rules to remember with regards to white space in XML:

1. White space within the content of an element is important; that is, the XML processor will pass these characters to the application or user agent.
2. White space in attributes is normalized; that is, neighboring white spaces are reduced to a single space.
3. White space in between elements is ignored.

- XML Syntax Rules

XML has relatively straightforward, but very strict, syntax rules. A document that follows these syntax rules is said to be well-formed.

1. There must be one and only one document element.
2. Every open tag must be closed.
3. If an element is empty, it still must be closed.
  - Poorly-formed: <TAG>
  - Well-formed: <TAG></TAG>
  - Also well-formed: <TAG />
4. Elements must be properly nested:
  - Poorly-formed: <A><B></A></B>
  - Well-formed: <A><B></B></A>
5. Tag and attribute names are case sensitive.
6. Attribute values must be enclosed in single or double quotes.

- Special Characters

There are five special characters that cannot be included in XML documents. These characters are replaced with predefined entity references as shown in the table below:

Character	Entity Reference
<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos;



3. Comments or Processing Instructions (optional)

Comments can appear throughout an XML document. Like in HTML, they begin with <!-- and end with -->.

```
<!--This is a comment-->
```

Processing instructions are used to pass parameters to an application. These parameters tell the application how to process the XML document. For example, the following processing instruction tells the application that it should transform the XML document using the XSL stylesheet artist.xsl

```
<?XML-stylesheet HREF="artist.xsl"
```

```
TYPE="text/xsl"?>
```

As shown above, processing instructions begin with and <? end with ?>.

**Let us see the progress:**

3. DTD has how many roles in XML document
  - a. 2
  - b. 3
  - c. 4
  - d. 1
  
4. White space is a prolog.
  - a. True
  - b. False
  
5. Which section of XML document will not be parsed by XML parser?
  - a. CDATA
  - b. PCDATA
  - c. DATA
  - d. CPDATA

---

### 9.3. CREATING XML DOCUMENTS

---

The following is relatively simple XML file describing the Artists:

**Artists.xml**

```
<?XML VERSION="1.0"?>
<ARTISTS>
  <ARTIST SITE="http://www.rajmehra.com">
    <NAME>
      <FIRSTNAME>Raj</FIRSTNAME>
      <LASTNAME>Mehra</LASTNAME>
    </NAME>
  </ARTIST>
  <ARTIST SITE="http://www.ajayverma.com">
    <NAME>
      <FIRSTNAME>Ajay</FIRSTNAME>
      <LASTNAME>Verma</LASTNAME>
    </NAME>
  </ARTIST >
  <ARTIST SITE="http://www.kapilsharma.com">
    <NAME>
```

```

<FIRSTNAME>Kapil</FIRSTNAME>
<LASTNAME>Sharma</LASTNAME>
</NAME>
</ARTIST >
<ARTIST SITE="http://www.rajivmalani.com"
NATIONAL="no">
<NAME>
<FIRSTNAME>Rajiv</FIRSTNAME>
<LASTNAME>malani</LASTNAME>
</NAME>
</ARTIST>
</ARTISTS>

```

### **Code Explanation**

In above document root element is ARTISTS. ARTISTS' element has one child element ARTIST. ARTIST also has one child element NAME. NAME has two child elements FIRSTNAME and LASTNAME. We can also observe that ARTIST element has two attributes SITE and NATIONAL.

---

## **9.4. XML DTDs**

---

A Document Type Definition (DTD) is a type of schema. The purpose of DTDs is to provide a framework for validating XML documents. By defining a structure that XML documents must conform to, DTDs allow different organizations to create shareable data files.

### **Well-formed vs. Valid**

1. A well-formed XML document is one that follows the syntax rules described in "XML Syntax Rules".
2. A valid XML document is one that conforms to a specified structure.
3. For an XML document to be validated, it must be checked against a schema (document that defines the structure for a class of XML documents).
4. XML documents that are not intended to conform to a schema can be well-formed, but they cannot be valid.

### **Creating DTDs**

DTDs are simple text files that can be created with any basic text editor. A DTD outlines what elements can be in an XML document and the attributes and sub-elements that they can take. Let's start by taking a look at a complete DTD and then dissecting it.

**Artists.dtd**

```

<!ELEMENT ARTISTS (ARTIST+)>
<!ELEMENT ARTIST (NAME)>
<!ATTLIST ARTIST
  SITE CDATA #IMPLIED
  NATIONAL (yes|no) "yes">
<!ELEMENT NAME (FIRSTNAME, LASTNAME)>
<!ELEMENT FIRSTNAME (#PCDATA)>
<!ELEMENT LASTNAME (#PCDATA)>

```

**The Document Element**

When creating a DTD, the first step is to define the document element.

```
<!ELEMENT ARTISTS (ARTIST+)>
```

The element declaration above states that the ARTISTS element must contain one or more ARTIST elements.

**Child Elements**

When defining child elements in DTDs, you can specify how many times those elements can appear by adding a modifier after the element name. If no modifier is added, the element must appear once and only once. The other options are shown in the table below:

Modifier	Description
?	Zero or One Times.
+	One or More Times.
*	Zero or More Times.

**Other Elements**

The other elements are declared in the same way as the document element - with the <!ELEMENT> declaration. The ARTISTS DTD declares four additional elements.

Each ARTIST element must contain a child element NAME, which must appear once and only once.

```
<!ELEMENT ARTIST (NAME)>
```

Each NAME element must contain a FIRSTNAME and LASTNAME element, which each must appear once and only once and in that order.

```
<!ELEMENT NAME (FIRSTNAME, LASTNAME)>
```

Some elements contain only text. This is declared in a DTD as #PCDATA. PCDATA stands for parsed character data, meaning that the data will be parsed for XML tags and entities. The FIRSTNAME and LASTNAME elements contain only text.

```

<!ELEMENT FIRSTNAME (#PCDATA)>
<!ELEMENT LASTNAME (#PCDATA)>

```

**Choice of Elements**

It is also possible to indicate that one of several elements may appear as a child element. E.g., the declaration below indicates that an IMG element may have a child element NAME or a child element ID, but not both.

```
<!ELEMENT IMG (NAME|ID)>
```

**Empty Elements**

Empty elements are declared as follows.

```
<!ELEMENT IMG EMPTY>
```

**Mixed Content**

Sometimes elements can have elements and text mixed. E.g., the following declaration is for a BODY element that may contain text in addition to any number of LINK and IMG elements.

```
<!ELEMENT BODY (#PCDATA | LINK | IMG)*>
```

**Location of Modifier**

The location of modifiers in a declaration is important. If the modifier is outside of a set of parentheses, it applies to the group; whereas, if the modifier is immediately next to an element name, it applies only to that element.

**The following examples illustrate:**

```
<!ELEMENT BODY (LINK* | IMG*)>
```

the BODY element can have any number of child LINK and IMG elements, but they must come in pairs, with the LINK element preceding the IMG element

```
<!ELEMENT BODY (LINK, IMG)*>
```

the BODY element can have any number of child LINK and IMG elements, but they must come in pairs, with the LINK element preceding the IMG element

```
<!ELEMENT body (link*, img*)>
```

the BODY element can have any number of child LINK elements followed by any number of child IMG elements

**Using Parentheses for Complex Declarations**

Element declarations can be more complex than the examples above. E.g., you can specify that a PERSON element either contains a single NAME element or a FIRSTNAME and LASTNAME element. To group elements, put them in parentheses as shown below:

```
<!ELEMENT PERSON (NAME | (FIRSTNAME, LASTNAME))>
```

**Declaring Attributes**

Attributes are declared using the <!ATTLIST > declaration. The syntax is shown below:

```
<!ATTLIST ElementName
AttributeName AttributeType State DefaultValue?
AttributeName AttributeType State DefaultValue?>
```

- ElementName is the name of the element taking the attributes.
- AttributeName is the name of the attribute.
- AttributeType is the type of data that the attribute value may hold. Although there are many types, the most common are CDATA (unparsed character data) and ID (a unique identifier). A list of options can also be given for the attribute type.
- DefaultValue is the value of the attribute if it is not included in the element.
- State can be one of three values: #REQUIRED, #FIXED (set value), and #IMPLIED (optional).

The ARTIST element has two possible attributes: SITE, which is optional and may contain any valid XML text, and NATIONAL, which defaults to yes if it is not included.

```
<!ATTLIST ARTIST
SITE CDATA #IMPLIED
NATIONAL (yes|no) "yes">
```

### **Validating an XML Document with a DTD**

The DOCTYPE declaration in an XML document specifies the DTD to which it should conform. In the code sample below, the DOCTYPE declaration indicates the file should be validated against artists.dtd in the same directory. Add below line in Artists.xml after declaration:

```
<!DOCTYPE ARTISTS SYSTEM "artists.dtd">
```

Below is the example how to work with internal and external DTD:

#### **Internal DTD**

```
<?XML VERSION="1.0"?>
<!DOCTYPE NOTE [
  <!ELEMENT NOTE (TO, FROM, HEADING, BODY)>
  <!ELEMENT TO (#PCDATA)>
  <!ELEMENT FROM (#PCDATA)>
  <!ELEMENT HEADING (#PCDATA)>
  <!ELEMENT BODY (#PCDATA)>
]>
<NOTE>
<TO>Amar</TO>
<FROM>Ajit</FROM>
<HEADING>Reminder</HEADING>
<BODY>Don't forget to read this</BODY>
</NOTE>
```

**External DTD**

This is the same XML document with an external DTD:

```
<?XML VERSION="1.0"?>
<!DOCTYPE NOTE SYSTEM "note.dtd">
<NOTE>
<TO>Amar</TO>
<FROM>Ajit</FROM>
<HEADING>Reminder</HEADING>
<BODY>Don't forget to read this</BODY>
</NOTE>
```

This is a copy of the file "note.dtd" containing the Document Type Definition:

```
<?XML VERSION="1.0"?>
<!ELEMENT NOTE (TO, FROM, HEADING, BODY)>
<!ELEMENT TO (#PCDATA)>
<!ELEMENT FROM (#PCDATA)>
<!ELEMENT HEADING (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
```

---

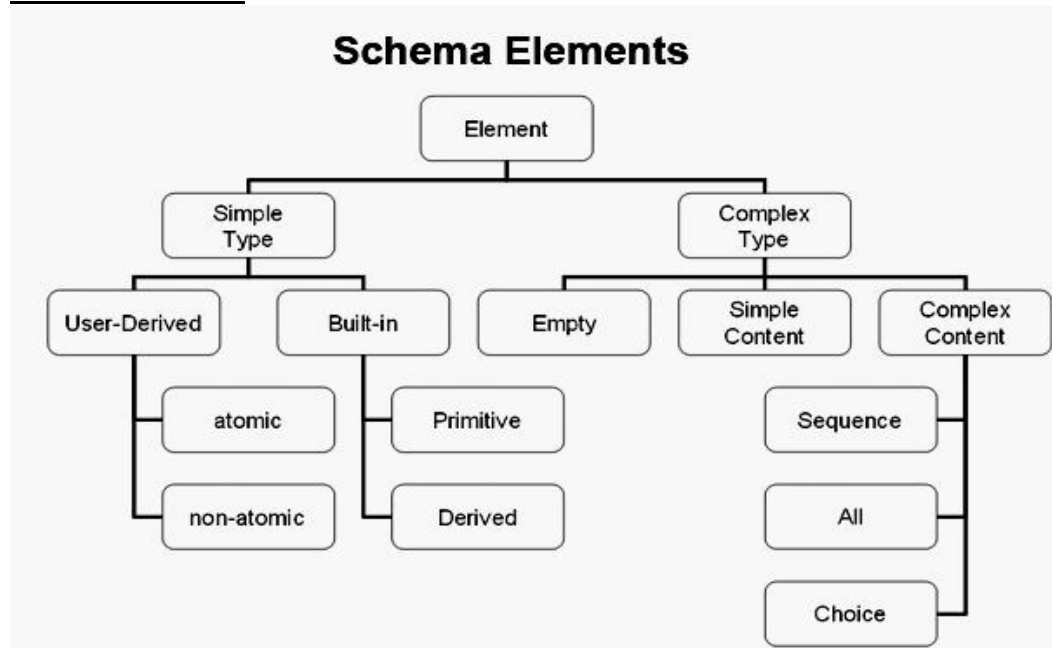
## **9.5. XML SCHEMAS**

---

It is an XML-based language used to create XML-based languages and data models. It defines element and attribute names for a class of XML documents. It specifies the structure that those documents must adhere to and the type of content that each element can hold.

### **Why need XML Schema / Limitations of DTDs**

- DTDs do not have built-in data types.
- DTDs do not support user-derived data types.
- DTDs allow only limited control over cardinality (the number of occurrences of an element within its parent).
- DTDs do not support Namespaces or any simple way of reusing or importing other schemas.

Schema Elements

Schema authors can define their own types or use the built-in types. The following is a high-level overview of Schema types. Elements can be of simple type or complex type.

## 1) Simple Type

- a) These elements can only contain text. They cannot have child elements or attributes.
- b) All the built-in types are simple types (E.g., XS:STRING).
- c) Schema authors can derive simple types by restricting another simple type. E.g., an email type could be derived by limiting a string to a specific pattern (that includes '@', '.', '\_', etc.)
- d) Simple types can be atomic (E.g., strings and integers) or non-atomic (E.g., lists).

## 2) Complex Type

- a) These elements can contain child elements and attributes as well as text.
- b) By default, complex-type elements have child elements.
- c) These elements can only contain text. But they are different from simple type elements in that they have attributes.
- d) These elements can be empty, but they may have attributes.
- e) These elements may have mixed content - a combination of text and child elements.

<p><u>Simple XML Schema – Student.xsd</u></p> <pre>&lt;?XML VERSION="1.0" ENCODING="UTF-8"?&gt; &lt;XS:SCHEMA XMLNS:XS="http://www.w3.org/2001/ XMLSchema"&gt;   &lt;XS:ELEMENT NAME="Student"&gt;     &lt;XS:COMPLEXTYPE&gt;       &lt;XS:SEQUENCE&gt;         &lt;XS:ELEMENT NAME="FirstName" TYPE="xs:string" /&gt;         &lt;XS:ELEMENT NAME="LastName" TYPE="xs:string" /&gt;       &lt;/XS:SEQUENCE&gt;     &lt;/XS:COMPLEXTYPE&gt;   &lt;/XS:ELEMENT&gt; &lt;/XS:SCHEMA&gt;</pre>	<p><u>The code below shows a valid XML instance of this XML schema – student1.xml</u></p> <pre>&lt;?XML VERSION="1.0"?&gt; &lt;STUDENT XMLNS:XSI="http://www.w3.org/ 2001/XMLSchema-instance" XSI:NONAMESPACESCHEMAL OCATION="Student.xsd"&gt; &lt;FIRSTNAME&gt;Sumit&lt;/FIRSTNA ME&gt; &lt;LASTNAME&gt;Tiwari&lt;/LASTNA ME&gt; &lt;/STUDENT&gt;</pre>
<p><u>Code Explanation:</u></p> <p>An XML schema is an XML document and must be well formed (i.e. follow all the syntax rules of XML document). XML schemas also have to follow the rules defined in the "Schema of schemas," which defines, among other things, the structure of an element and attribute names in an XML schema. It is a common practice to use the XS qualifier to identify Schema elements and types.</p> <p>The document element of XML schemas is XS:SCHEMA. It takes the attribute XMLNS:XS with the value of http://www.w3.org/2001/XMLSchema, indicating that the document should follow the rules of XML Schema.</p> <p>In this XML schema, we see a XS:ELEMENT element within the XS:SCHEMA element. XS:ELEMENT is used to define an element. In this case it defines the element STUDENT as a complex type element, which contains a sequence of two elements: FIRSTNAME and LASTNAME, both of which are of the simple string type.</p>	<p><u>Code Explanation:</u></p> <p>This is a simple XML document. Its document element is STUDENT, which contains two child elements: FIRSTNAME and LASTNAME, just as the associated XML schema requires. The XMLNS:XSI attribute of the document element indicates that this XML document is an instance of an XML schema. The document is tied to a specific XML schema with the XSI:NONAMESPACESCHEMALOCATION attribute.</p>



---

## 9.6. XSL

---

HTML pages use predefined tags, and the meaning of these tags is well understood: <P> means a paragraph and <H1> means a header, and the browser knows how to display these pages.

With XML we can use any tags we want and the meaning of these tags are not automatically understood by the browser: <TABLE> could mean a HTML table or maybe a piece of furniture. Because of the nature of XML, there is no standard way to display an XML document.

In order to display XML documents, it is necessary to have a mechanism to describe how the document should be displayed. One of these mechanisms is Cascading Style Sheets (CSS), but XSL (eXtensible Stylesheet Language) is the preferred style sheet language of XML, and XSL is far more sophisticated than the CSS used by HTML. XSL consists of two parts:

- a method for transforming XML documents (XSLT)
- a method for formatting XML documents (XSL-FO)

XSL is a language that can transform XML into HTML, a language that can filter and sort XML data and a language that can format XML data, based on the data value, like displaying negative numbers in red.

### XSLT

An XSLT looks at an XML document as a collection of nodes of the following types:

- Root node
- Element nodes
- Attribute nodes
- Text nodes
- Processing instruction nodes
- Comment nodes

An XSLT document contains one or more templates, which are created with the

<XSL:TEMPLATE /> tag. The XSLT processor reads through the XML document starting at the root, progressing from top to bottom.

<u>Example – artists.xml</u>	<u>Code Explanation</u>
<pre>&lt;?XML VERSION="1.0"?&gt; &lt;XSL:STYLESHEET VERSION="1.0" XMLNS:XSL="http://www.w3.org/1999/ XSL/Transform"&gt; &lt;XSL:OUTPUT METHOD="html"/&gt; &lt;XSL:TEMPLATE MATCH="child::ARTIST"&gt;</pre>	<p>Document begins with an XML declaration. As with all XML documents, the XML declaration is optional.</p> <p>The second line is the document element of the XSLT. It states that this document is a version 1.0</p>

<pre> &lt;HTML&gt; &lt;HEAD&gt; &lt;TITLE&gt; &lt;XSL:VALUE-OF SELECT="descendant::FIRSTNAME" /&gt; &lt;XSL:TEXT&gt; &lt;/XSL:TEXT&gt; &lt;XSL:VALUE-OF SELECT="descendant::LASTNAME" /&gt; &lt;/TITLE&gt; &lt;/HEAD&gt; &lt;BODY&gt; &lt;TABLE BORDER="1" WIDTH="200"&gt; &lt;TR&gt;&lt;TD&gt; &lt;XSL:VALUE-OF SELECT="descendant::FIRSTNAME" /&gt; &lt;XSL:TEXT&gt; &lt;/XSL:TEXT&gt; &lt;XSL:VALUE-OF SELECT="descendant::LASTNAME" /&gt; &lt;/TD&gt; &lt;/TR&gt; &lt;/TABLE&gt; &lt;/BODY&gt; &lt;/HTML&gt; &lt;/XSL:TEMPLATE&gt; &lt;/XSL:STYLESHEET&gt; </pre>	<p>XSLT document.</p> <pre> &lt;XSL:STYLESHEET VERSION="1.0" XMLNS:XSL="http://www.w3.org/1999/XSL/Transform"&gt; </pre> <p>The third line indicates that the resulting output will be HTML.</p> <pre> &lt;XSL:OUTPUT METHOD="html" /&gt; </pre> <p>The fourth line is an open <code>&lt;XSL:TEMPLATE&gt;</code> element. The <code>MATCH</code> attribute of this tag takes an XPath, which indicates that this template applies to the <code>ARTIST</code> node of the XML document. Because <code>ARTIST</code> is the document element of the source document, this template will only run once.</p> <p>There are then a few lines of HTML tags followed by two <code>&lt;XSL:VALUE-OF /&gt;</code> elements separated by one <code>&lt;XSL:TEXT&gt;</code> element. The <code>&lt;XSL:VALUE-OF /&gt;</code> tag has a <code>SELECT</code> attribute, which takes an XPath pointing to a specific element or group of elements within the XML document. In this case, the two <code>&lt;XSL:VALUE-OF /&gt;</code> tags point to <code>FIRSTNAME</code> and <code>LASTNAME</code> elements, indicating that they should be output in the title of the HTML page. The <code>&lt;XSL:TEXT&gt;</code> element is used to create a space between the <code>FIRSTNAME</code> and the <code>LASTNAME</code> elements.</p> <pre> &lt;XSL:VALUE-OF SELECT="descendant::FIRSTNAME" /&gt; &lt;XSL:TEXT&gt; &lt;/XSL:TEXT&gt; &lt;XSL:VALUE-OF SELECT="descendant::LASTNAME" /&gt; </pre> <p>There are then some more HTML tags followed by the same XSLT tags, re-outputting the <code>FIRSTNAME</code> and <code>LASTNAME</code> of the Beatle in the body of the HTML page.</p>
---	--

After creating artists.xml place it in the same directory as of artists.xml. Also insert the following line in artists.xml after declaration:

```
<?XML-STYLE SHEET HREF="artists.xsl" TYPE="text/xsl"?>
```

Save artists.xml and open it in browser. You will be able to find the output in tabular format as shown below:

Raj Mehra
Ajay Verma
Kapil Sharma
Rajiv malani

---

## 9.7. SUMMARY

---

In this chapter we have studied the purpose of XML documents, how to create a simple xml document, how to include style sheets (XSL) in xml document to render the data in tabular way. We also studied the role of DTD in xml documents, when a document can be called well-formed/valid

Answer to let's check your progress

1. c                      2. c                      3. b                      4. a                      5. a

---

## 9.8. EXERCISE

---

### 9.8.1. Questions

1. What is XML? How it is different from HTML?
2. Explain benefits of XML.
3. What is format of XML Document?
4. Explain document element, empty elements, and attributes in XML.
5. Explain CDATA. Also explain how whitespaces are handled in XML.
6. State and explain syntax for forming a well formed XML document.
7. How to include special characters like '<', '>', '&', '"', ''' in XML document.
8. Explain the difference between well formed and valid XML document.

9. Explain role of modifier in DTDs.
10. Write and explain syntax for declaring attributes in DTDs.
11. What is XML Schema? Why we require it?
12. Explain Schema element structure.
13. What is XSL? Why we need it.
14. Explain the XSL file in detail (to be created in program 3 below).

### **9.8.2. Programs**

1. Create an XML document that will store information about vehicle i.e. color of vehicle, manufacturing company (this will have information such as Indian company or foreign company), year of manufacturing.
2. Create DTD for the XML document created in program 1.
3. Create an XSL file for the xml document created in program 1.



## PHP

### Unit Structure

- 10.0. Objective
- 10.1. Why PHP and MYSQL?
- 10.2. Server-Side Web Scripting
- 10.3. Installing PHP
- 10.4. Adding PHP to HTML
- 10.5. Syntax and Variables
- 10.6. Passing Information between Pages
- 10.7. Strings
- 10.8. Arrays and Array Functions
- 10.9. Numbers
- 10.10. Basic PHP Errors Problems
- 10.11. Exercise

---

### 10.0. OBJECTIVE

---

After going through this chapter you will be able to:

- Identify why to use PHP
- Explain how to install Apache webserver
- Explain how to install PHP
- Explain how to install MySQL
- Explain how to install PHPMyAdmin
- Identify role of php.ini and working of PHP
- Identify different methods of displaying text using PHP
- Identify how to create variable in PHP and what are different data types of variables

---

### 10.1. WHY PHP AND MYSQL?

---

PHP stands for **PHP** Preprocessor **Hy**PerText. It is a server-side scripting language, like ASP. All PHP scripts are executed on the server. It supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.). It is open source software that can be freely downloaded from [www.php.net](http://www.php.net) and used.

A PHP file has a file extension of ".php", ".php3", or ".phtml" and contains text, HTML tags and scripts that are returned to the browser as plain HTML.

PHP runs on different platforms (i.e., Windows, LINUX, UNIX, etc.) and is compatible with almost all servers used today including APACHE, IIS, etc. It is easy to learn and runs efficiently on the server side.

MySQL is a free to download and use database server that is ideal for both small and large applications. It supports standard SQL and compiles on a number of platforms.

PHP and MySQL when combined together are cross-platform, i.e., you can develop in Windows and serve on a UNIX platform.

---

## 10.2. SERVER-SIDE WEB SCRIPTING

---

---

## 10.3. INSTALLING PHP

---

To run PHP files we require web server as mentioned earlier. It can be either APACHE or IIS.

Where to download:

- **Download PHP:** <http://www.php.net/downloads.php>
- **Download MySQL Database:** <http://www.mysql.com/downloads/>
- **Download Apache Server:** <http://httpd.apache.org/download.cgi>

In this chapter four installation procedures are shown:

1. For Apache Web Server
2. For PHP
3. For MySQL
4. For PHPMyAdmin

To work with PHP we have to install PHP and a Web Server. The best combination of such type is PHP and APACHE so installation of APACHE Web Server is shown. If you are using Windows XP then IIS will be installed on computer. Microsoft's product IIS is also a Web Server. So you can configure IIS (shown in PHP Installation) to work with PHP. As already discussed, combination of PHP and MySQL is the best. So for database interaction using PHP we have to install MySQL. In MySQL when you want to create database and tables or insert values, etc., you need an interface to do this directly in backend so PHPMyAdmin help you to do that.

### Installation of APACHE

Download the best available version of Apache Web Server for Windows from this <http://httpd.apache.org/download.cgi>

#### Select:

Win32 Binary without crypto (no mod\_ssl) (MSI Installer)

#### Or

Win32 Binary including OpenSSL

Here we are using APACHE HTTP Server (httpd) 2.2.17 Win32 Binary including OpenSSL

Double click the installer and then click NEXT. Check "I accept the terms in the license agreement" and click NEXT twice. Now you will see the below image to fill the server information:

**Apache HTTP Server 2.2 - Installation Wizard**

**Server Information**

Please enter your server's information.

Network Domain (e.g. somenet.com)  
localhost

Server Name (e.g. www.somenet.com):  
localhost

Administrator's Email Address (e.g. webmaster@somenet.com):  
admin@localhost.com

Install Apache HTTP Server 2.2 programs and shortcuts for:

For All Users, on Port 80, as a Service -- Recommended.

only for the Current User, on Port 8080, when started Manually.

InstallShield

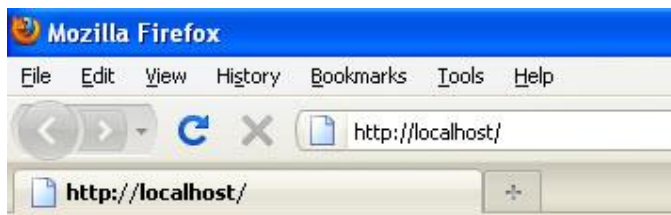
< Back    Next >    Cancel

After filling the information as given above click NEXT three times. It will start the installation process after you click INSTALL. Once the installation is complete you must click FINISH.

After finishing, open your favorite browser and type the following in the address bar and press the ENTER key:

`http://localhost/`

If you will see 'It works!' , then your apache works fine.



**It works!**

### Installation of PHP

It is assumed that you have already successfully setup IIS or installed APACHE on your machine and configured it. So, to configure PHP and secure its correct operation you need to go through a few steps:

1. Download and unzip the latest version of PHP

Download the latest zipped distribution of PHP from <http://php.net>. Unzip it. The recommendation is to put all the PHP stuff in a folder just off of the root drive (avoid whitespace), like C:\PHP.

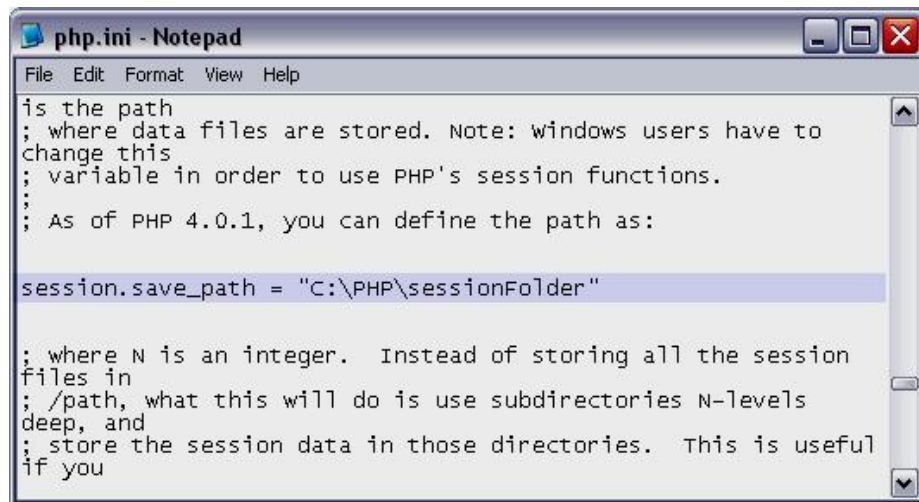
2. Rename/copy php.ini-recommended to php.ini

In your PHP directory, you'll find a couple of php.ini-\* files. They are pre-configured settings for a PHP install that you can use as an initial setup. The php.ini-recommended is the most secure; hence, the recommended one; just rename it to php.ini and copy to your Windows directory.

3. Create a session state directory and point the session.save\_path variable in php.ini to it

This is optional, but recommended step. PHP does not need sessions, but it's something that will most likely be useful. Create a session directory somewhere on the server. I created C:\PHP\sessionFolder. This directory will hold many small files with session variable information for PHP. Now change the value of the session.save\_path variable in php.ini to be the full path to that directory (session.save\_path=C:\PHP\sessionFolder).





```

File Edit Format View Help

is the path
; where data files are stored. Note: windows users have to
change this
; variable in order to use PHP's session functions.
;
; As of PHP 4.0.1, you can define the path as:

session.save_path = "C:\PHP\sessionFolder"

; where N is an integer. Instead of storing all the session
files in
; /path, what this will do is use subdirectories N-levels
deep, and
; store the session data in those directories. This is useful
if you

```

#### 4. Setup the PHP extensions

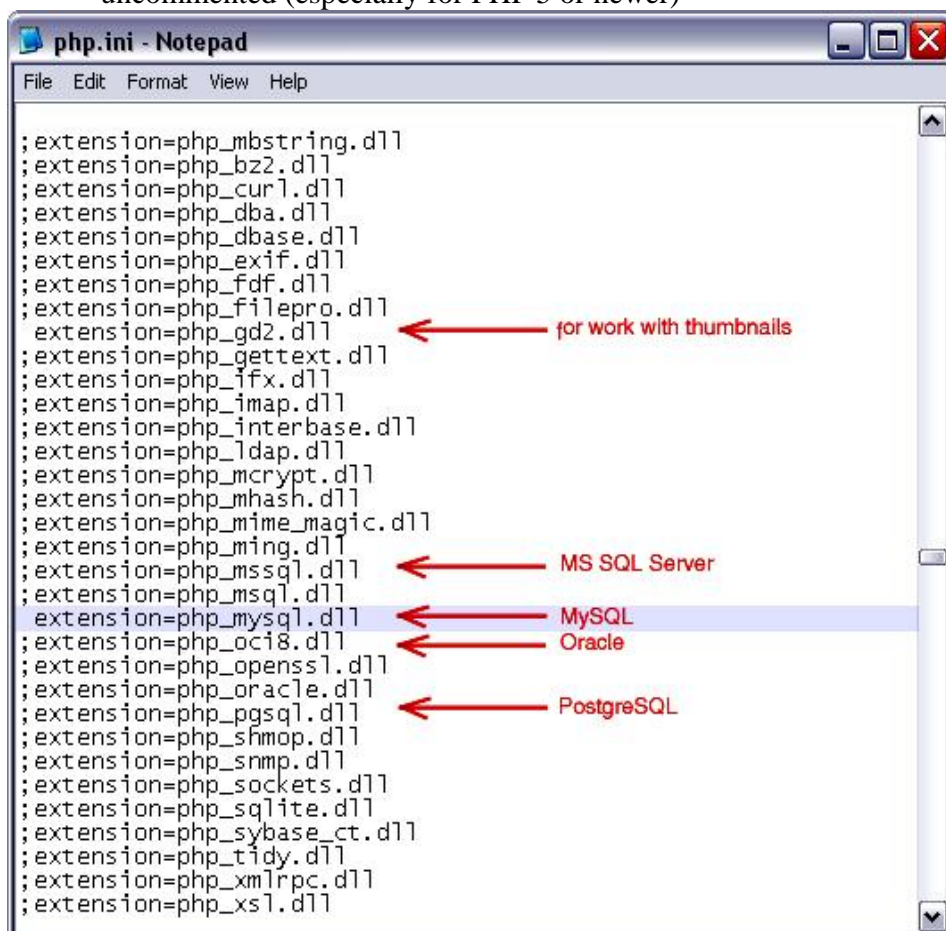
You need to point PHP to the directory that holds the extension libraries and you need to uncomment the desired extensions.

Point PHP to the correct directory:

Set `extension_dir` in `php.ini` to `"C:\PHP\ext"` (`extension_dir = "C:\PHP\ext"`).

Uncomment the ones you want to use:

It is important to be sure that `php_mysql.dll` extension is uncommented (especially for PHP 5 or newer)

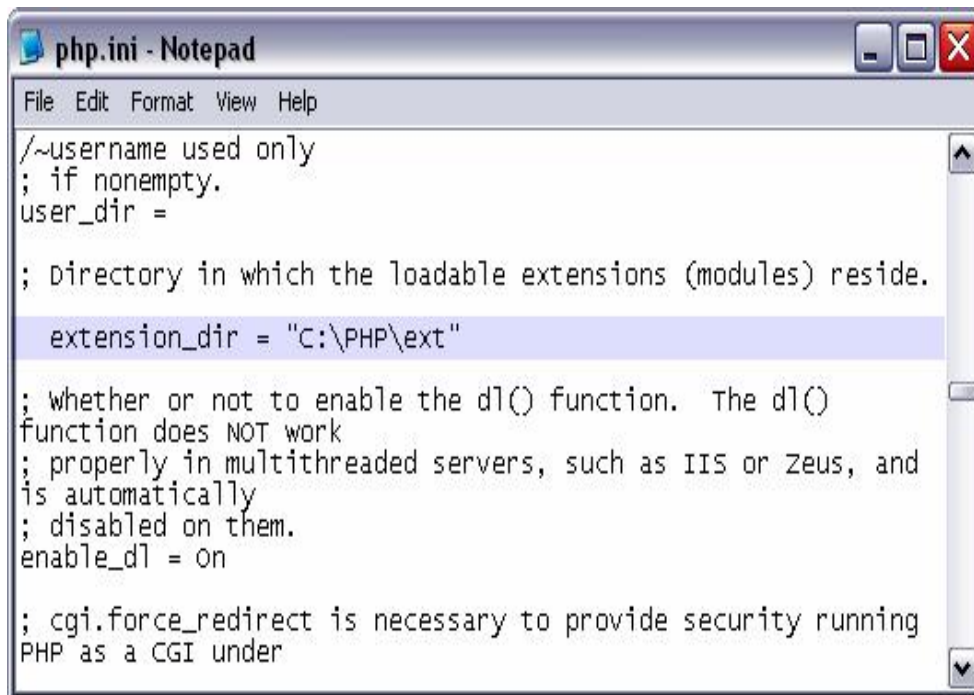


```

File Edit Format View Help

;extension=php_mbstring.dll
;extension=php_bz2.dll
;extension=php_curl.dll
;extension=php_dba.dll
;extension=php_dbase.dll
;extension=php_exif.dll
;extension=php_fdf.dll
;extension=php_filepro.dll
extension=php_gd2.dll ← for work with thumbnails
;extension=php_gettext.dll
;extension=php_ifx.dll
;extension=php_imap.dll
;extension=php_interbase.dll
;extension=phpldap.dll
;extension=php_mcrypt.dll
;extension=php_mhash.dll
;extension=php_mime_magic.dll
;extension=php_ming.dll
;extension=php_mssql.dll ← MS SQL Server
;extension=php_mysql.dll ← MySQL
;extension=php_oci8.dll ← Oracle
;extension=php_openssl.dll
;extension=php_oracle.dll
;extension=php_pgsql.dll ← PostgreSQL
;extension=php_shmop.dll
;extension=php_snmp.dll
;extension=php_sockets.dll
;extension=php_sqlite.dll
;extension=php_sybase_ct.dll
;extension=php_tidy.dll
;extension=php_xmlrpc.dll
;extension=php_xsl.dll

```



```

File Edit Format View Help
/~username used only
; if nonempty.
user_dir =

; directory in which the loadable extensions (modules) reside.
extension_dir = "C:\PHP\ext"

; whether or not to enable the dl() function. The dl()
function does NOT work
; properly in multithreaded servers, such as IIS or Zeus, and
is automatically
; disabled on them.
enable_dl = on

; cgi.force_redirect is necessary to provide security running
PHP as a CGI under

```

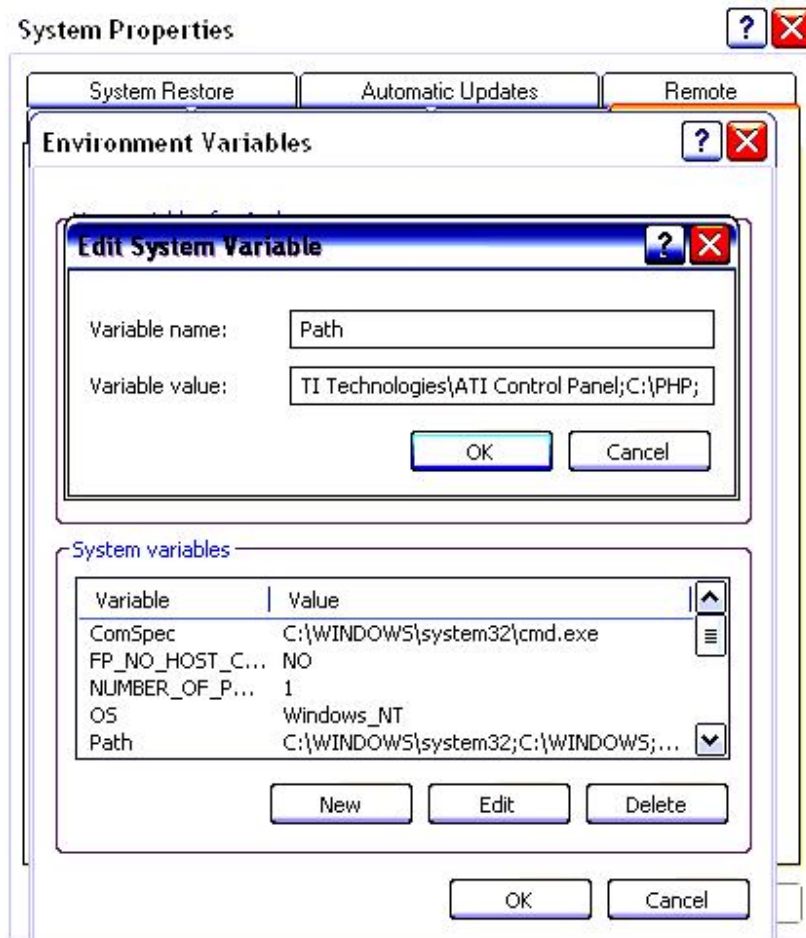
5. Make sure that PHP folder is in the system path

You should add "C:\PHP" to the server's PATH environment variable. To do so follow the steps given below:

- Right-click on My Computer, choose Properties
- Flip to the Advanced tab
- Click the Environment Variables button
- Double-click the Path variable in the list of System variables.
- Either add "C:\PHP;" to the beginning or ";C:\PHP" to the end (sans quotes, not both).
- Restart IIS for it to take effect.

(To restart IIS you should right-click the local computer in the left pane of IIS Manager, click on All Tasks -> Restart IIS... -> OK)

Instead, you can copy all non-php dll files from C:\PHP to C:\Windows\System32 (or somewhere else in the server's PATH), but the first is the preferred method since it keeps the installation in one place, making upgrading or uninstalling easier.



## 6. Configure IIS

For configuration, open IIS Manager (Start -> Control Panel -> Administrative Tools -> Internet Information Services (IIS) Manager). Then add new extension (.php). Expand the local computer in the left pane. Right-click on "Web Sites" in the left pane, then click "Properties" in the menu that pops up. Flip top the Home Directory tab and click "Configuration". Flip to the Mappings tab and click Add. Enter the full path to php5isapi.dll in the "Executable" textbox (Browse... to find it more easily if you need to). Enter ".php" in the Extension textbox. Select radial button Limit to, enter "GET, POST, HEAD". Click OK all the way out.

This will apply to every website. This sets up IIS to actually respond to requests for PHP files. Until now, IIS did not know what to do with PHP files, you just told it to pass them through php5isapi.dll.

## 7. Configure Apache Web Server

If you want PHP to work with your APACHE Web Server, you will need to modify your APACHE configuration file to load it.

There are two ways to configure APACHE to use PHP. One is to configure it to load the PHP interpreter as an APACHE module. The other is to configure it to run the PHP interpreter as a CGI binary. Unless you have a particular reason for running PHP as a CGI binary, you will probably want to load PHP as a module in APACHE, since it runs more efficiently that way.

To configure APACHE to load PHP as a module to parse your PHP scripts you should make some changes in the APACHE configuration file, "httpd.conf", typically found in "c:\Program Files\Apache Group\Apache\conf\". It also can be accessed from your program files menu.

Search for the section that has a series of commented out "LoadModule" statements. Add the following line after all the LoadModule statements:

```
LoadModule php5_module "c:/php/php5apache2.dll"
```

Search for "AddType" and add the following line after the last "AddType" statement:

```
AddType application/x-httpd-php .php
```

If you need to support other file types, like ".php3" and ".phtml", simply add them to the list, like this:

```
AddType application/x-httpd-php .php3
```

```
AddType application/x-httpd-php .phtml
```

#### 8. Test your setup

Create a new file named "test.php" in one of the websites. Expand the Web Sites folder in the left pane of IIS Manager to see a list of existing websites. Right-click on a website and click on Properties -> Home Directory -> Local Path. It will show you where the website root directory is.

Create test.php file with the following line:

```
<?php phpinfo(); ?>
```

With your browser go to <http://localhost/test.php>

After loading test.php, you should see some information about your PHP installation. Be sure to scroll all the way down to the bottom to see that there were no errors. Pay attention to "Configuration File (php.ini) Path" field. Field's value is current location of php.ini file and you should make appropriate changes in it.

#### 9. Location of libmysql.dll

Lastly, we need to be sure that copy of libmysql.dll is located in PHP folder (for PHP 5.0 or newer).

### Installation of MySQL

Download the latest MySQL version from this  
<http://www.mysql.com/downloads/>

Click MySQL Community Server and select Windows (x86, 32-bit), MSI Installer to download. Here we are using MySQL Community Server 5.1.51

Double click the installer and click NEXT twice. Then click on INSTALL and NEXT twice.



After the above, click NEXT ten times.



If this is your first MySQL Database setup, you simply enter your root password. After you enter your root password it will enable the NEXT button. Click on NEXT then EXECUTE and then FINISH.

To test MySQL, open your MySQL command line client (START->All Programs-> MySQL-> MySQL Server 5.1-> MySQL Command Line Client). Type your password and you will get like this:

```

mysql> Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.51-community MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

```

### Installing phpMyAdmin

phpMyAdmin is a free software tool written in PHP intended to handle the administration of MySQL over the World Wide Web. phpMyAdmin

supports a wide range of operations with MySQL. The most frequently used operations are supported by the user interface (managing databases, tables, fields, relations, indexes, users, permissions, etc), while you still have the ability to directly execute any SQL statement.

Download the latest phpMyadmin Zip version from this:

[http://www.phpmyadmin.net/home\\_page/downloads.php](http://www.phpmyadmin.net/home_page/downloads.php)

Here we are using phpMyAdmin 3.3.8. Extract the zip archive to this location:

C:\Program Files\Apache Software Foundation\Apache2.2\htdocs

And rename the phpMyAdmin-3.3.8-all-languages folder to phpmyadmin. Here 3.3.8 is version number. Now open your apache httpd.conf file and find these lines:

```
<IfModule dir_module>
DirectoryIndex index.html
</IfModule>
```

Add index.php after the index.html and it looks like this:

```
<IfModule dir_module>
DirectoryIndex index.html index.php
</IfModule>
```

Restart your apache server. Open your browser and type the following in the address bar and press ENTER:

<http://localhost/phpmyadmin/>



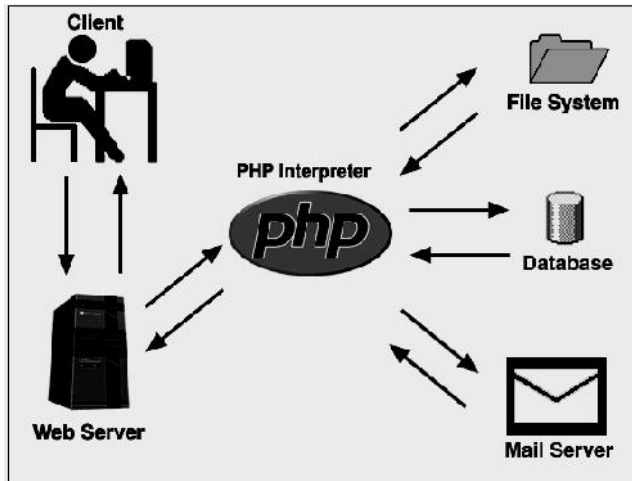
The screenshot shows the phpMyAdmin login interface. At the top, there is a logo with a sailboat and the text 'phpMyAdmin'. Below the logo, it says 'Welcome to phpMyAdmin'. There is a 'Language' dropdown menu set to 'English'. A 'Log in' section contains a 'Username' field with 'root' entered and a 'Password' field with four dots. A 'Go' button is located at the bottom right of the login section.

Type your username and password to login.

---

## 10.4. HOW PHP WORKS

---



As shown in the diagram above, the PHP interpreter processes the page, communicating with file systems, databases, and email servers as necessary, and then delivers a web page to the web server to return to the browser.

### Role of PHP.ini file

- A plain text file that is used to configure PHP
- When the PHP interpreter is started, it reads the php.ini file to determine what settings to use

---

## 10.5. PHP TAGS

---

PHP scripts are always enclosed in between two PHP tags. This tells your server to parse the information between them as PHP. The four different forms are as follows:

<code>&lt;?php PHP CODE GOES IN HERE ?&gt;</code>	This is the most commonly used (and recommended) form. It is known as the XML style, because it can be used inside of an XML document without causing the document to become poorly formed.
<code>&lt;script language="php"&gt; PHP CODE GOES IN HERE &lt;/script&gt;</code>	HTML or Script style tags.
<code>&lt;? PHP CODE GOES HERE ?&gt;</code>	"Short" tags
<code>&lt;% PHP CODE GOES HERE %&gt;</code>	ASP-style



### Creating and running Your First Script

The first PHP script you will be writing is very basic. All it will do is print out all information about PHP on your server. Type the following code into your text editor:

```
<?
phpinfo();
?>
```

As you can see this actually just one line of code. It is a standard PHP function called `phpinfo` which will tell the server to print out a standard table of information giving you information on the setup of the server.

One other thing you should notice in this example is that the line ends in a semicolon. This is very important. In PHP nearly all lines are ended with a semicolon and if you miss it out you will get an error.

Now you have finished your script save it as `phpinfo.php` and upload it to your server in the normal way. Now, using your browser, go the URL of the script. If it has worked (and if PHP is installed on your server) you should get a huge page full of the information about PHP on your server.

If your script doesn't work and a blank page displays, you have either mistyped your code or your server does not support this function (although I have not yet found a server that does not). If, instead of a page being displayed, you are prompted to download the file, PHP is not installed on your server and you should either search for a new web host or ask your current host to install PHP.

### Displaying Text

To display text using your PHP script is actually very simple. As with most other things in PHP, you can do it in a variety of different ways. The main one you will be using, though, is `print`. `print` will allow you to output text, variables or a combination of the two so that they display on the screen.

The `print` statement is used in the following way:

```
<?
print("Hello world!");
?>
```

#### Code Explanation

`print` is the command and tells the script what to do. This is followed by the information to be printed, which is contained in the brackets. Because you are outputting text, the text is also enclosed inside quotation marks. Finally, as with nearly every line in a PHP script, it must end in a semicolon. This command is enclosed in your standard PHP tags to declare it as PHP.

Which will display following on the screen.

```
Hello world!
```

Instead of using print(), we can also use echo() to display text on the page as follows:

```
<?
echo("Hello world!");
?>
```

**Let us check the progress:**

1. Which functions will display text on the page?
  - a. echo
  - b. print
  - c. scanf
  - d. display
  
2. In how many ways tags can be used in PHP?
  - a. 2
  - b. 3
  - c. 4
  - d. 5

---

## 10.6. VARIABLES

---

As with other programming languages, PHP allows you to define variables. In PHP there are several variable types, but the most common is called a String. It can hold text and numbers. All strings begin with a \$ sign. To assign some text to a string you would use the following code:

```
$text = "welcome to my website.";
```

This is quite a simple line to understand, everything inside the quotation marks will be assigned to the string. You must remember a few rules about strings:

1. Strings are case sensitive so \$Text is not the same as \$text
2. String names can contain letters, numbers and underscores but cannot begin with a number or underscore

When assigning numbers to strings you do not need to include the quotes so:

```
$emp_id = 456
```

would be allowed.

### Variable Types

Variable Type	Explanation
Integer	whole number
Double	real number
String	string of characters
Boolean	true or false
Array	list of items
Object	instance of a class

PHP is weakly typed, meaning that variables do not need to be assigned a type (E.g., Integer) at the time they are declared. Rather, the type of a PHP variable is determined by the value the variable holds and the way in which it is used.

E.g., We would first want to make a variable name and then set that equal to the value we want.

```
PHP Code:
<?php
$hello = "Hello World!";
$a_number = 4;
$anotherNumber = 8;
?>
```

Note: PHP does not require variables to be declared before being initialized.

### Variable Naming Conventions

There are a few rules that you need to follow when choosing a name for your PHP variables.

1. PHP variables must start with a letter or underscore "\_".
2. PHP variables may only be comprised of alpha-numeric characters and underscores. a-z, A-Z, 0-9, or \_ .
3. Variables with more than one word should be separated with underscores. \$my\_variable
4. Variables with more than one word can also be distinguished with capitalization. \$myVariable

### Echoing Variables

Echoing variables is very easy. No quotations are required, even if the variable does not hold a string. Below is the correct format for echoing a variable.

```
PHP Code:
<?php
$my_string = "Hello Boss. My name is: ";
$my_name = "Zubin";
$my_letter = 4;
echo $my_string;
echo $my_name;
echo $my_letter;
?>
```

### Display:

Hello Boss. My name is: Zubin 4

---

## 10.7. SUMMARY

---

In this chapter we have discussed what PHP is, what MySQL is, and Why to use PHP and MySQL. We have learnt how to install and configure PHP, MySQL, Apache, and PHPMyAdmin. We discussed to work PHP requires php.ini and what is its role, various methods are available to display text on the web page, how to create a variable, different data types a variable can take.

Answer to let's check your progress:

1. a, b
2. c

---

## 10.8. EXERCISE

---

### 10.8.1. Questions

1. What is PHP?
2. What is MySQL?
3. Why to use PHP and MySQL?
4. Explain how to install PHP and make it work with IIS?
5. Explain how PHP works? And what is the role of php.ini?
6. How to display text on the web page using PHP?
7. Write a note on variables in PHP.



## SAMPLE QUESTION PAPER 1

- Note:**
1. All questions are compulsory.
  2. Draw neat labeled diagrams and give examples wherever necessary.
  3. Figures to the right indicate full marks.

1. A. Explain <input> tag in detail 10
  
2. **(Attempt any three)**
  - A. Explain drawing diagram email system 5
  - B. Explain three technologies of e-commerce 5
  - C. Explain role of IP Address in networking. Also explain different classes of IP address. 5
  - D. Write note proxy server 5
  
3. **(Attempt any three)**
  - A. Explain with example any five global attributes 5
  - B. Explain <UL> and <LI> tags 5
  - C. Explain <FRAME> tag 5
  - D. What are Style sheets? Explain different approaches to style sheet 5
  
4. **(Attempt any three)**
  - A. Explain special operators of JavaScript – comma, delete, in, instanceof, new 5
  - B. Explain different Loops available in JavaScript 5
  - C. How to create Number object in JavaScript? Explain any four methods of Number object 5
  - D. What are regular expressions? Explain giving example 5

5. **(Attempt any three)**
- A. Explain benefits of XML 5
  - B. Explain syntax rules available in XML 5
  - C. Explain the “Well-formed vs. Valid XML document” 5
  - D. State and explain purpose of XML Schema 5
6. **(Attempt any three)**
- A. What is PHP? Why to use PHP and MySql 5
  - B. How to create associated arrays in PHP? Explain any four Array Functions 5
  - C. How error handling is done in PHP? Explain 5
  - D. Explain ‘.’, ‘?:’, ‘@’ operators of PHP 5
7. **(Attempt any three)**
- A. State and explain steps for querying a database in a PHP script 5
  - B. Write a note on Sessions in PHP 5
  - C. Write a note on type conversion in PHP. Give Example 5
  - D. Write a PHP script that will create table ‘Person’ (FirstName, LastName, Age) in database ‘mydb’ 5

